

ORACLE7 FOR SUN-SPARC SUNOS 4.1.x INSTALLATION & CONFIGURATION GUIDE

RELEASE 7.0.12

Part No. 7102-70-0193

January 29, 1993

ORACLE®

Cooperative Server Technology for Transparent Data Sharing

ORACLE7 FOR SUN-SPARC SUNOS 4.1.x INSTALLATION & CONFIGURATION GUIDE

RELEASE 7.0.12

Part No. 7102-70-0193

January 29, 1993

ORACLE®

Cooperative Server Technology for Transparent Data Sharing

ORACLE7 FOR SUN-SPARC SUNOS 4.1.X INSTALLATION & CONFIGURATION GUIDE,
Release. 7.0.12
Part No. 7102-70-0193-60-0992

Copyright © Oracle Corporation 1985, 1990, 1991, 1992. All rights reserved.
Printed in Ireland.

Contributing Authors: Tara Maria Bosco, Bill Branca, Donn Fisher, Sylvia Goldsmith, Laura Humphrey, Joanne Lai, Rita Moran, Jane Sokolow, Eileen Tichane, Paula White-Reddy, Chris Underwood

Contributors: Roshanak Afsarifar, Patrick Amor, Joan Arnett, Guy Binns, Florence Blangy, Tom Borja, Steven Bower, Jim Boyce, Terri Czerwinski, Henry Dai, Randy Eckstein, Andres Espineira, Ramana Gourinani, John Graham, Donna Hendrix, Salem Khan, Hrishikamat, Mike Kavanaugh, Qingbo Kong, Peter Konopka, Madhu Krishnapuram, Pei L. Ku, James Li, Cary Millsap, Brian Moriarty, Jatin Patel, Jeff Pau, Tow Ming Siow, Alan Tam, Tak Wang, Rick Wessman, Ray Wong, Henry Dai, T.S. Ramakrishnan, Joseph Ung, Sandip Nerkar, Harish Kurup, Stephen Peters, Bharat Vijayaraghavan

This documentation contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend: Use, duplication, or disclosure of the Programs by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with "Restricted Rights," as defined in FAR 52.227-14, Rights in Data - General, including Alternate III (June 1987).

ORACLE, Easy*SQL, SQL*Calc, SQL*Forms, SQL*Loader, SQL*Menu, SQL*Net, SQL*Plus, SQL*Report, Pro*Ada, Pro*COBOL, Pro*FORTRAN, and Pro*Pascal are registered trademarks of Oracle Corporation. ORACLE7, Pro*C, Oracle*Mail, Oracle*Terminal, PL/SQL, SQL*DBA, and SQL*ReportWriter are trademarks of Oracle Corporation.

Sun Workstation and Solaris are registered trademarks, and SunOS, Sun Microsystems, OpenWindows, X11/NeWS, and the combination of Sun with a numeric suffix are trademarks of Sun Microsystems, Inc. SPARC is a registered trademark of SPARC International, Inc. The X Window System is a trademark of the Massachusetts Institute of Technology. UNIX is a registered trademark, and OPEN LOOK is a trademark of UNIX System Laboratories, Inc.

CONTENTS

PART 1: INSTALLATION

CHAPTER 1	KEY TO ORACLE ON SUNOS	1-3
	System Requirements for ORACLE on SunOS	1-4
	Oracle Tools Requirements for ORACLE on SunOS	1-6
	Disk Space and Memory Requirements	1-13
	Shared Memory	1-22
	SunOS-Specific Issues	1-23
	SunOS-Specific Restrictions	1-28
	Restrictions and Work-arounds	1-31
	Product Obsolescence	1-36
 CHAPTER 2	 PREPARING TO INSTALL ORACLE PRODUCTS	 2-1
	Where to Begin	2-2
	Introduction to the Oracle Base Structure	2-4
	SunOS System Administrator	2-5
	ORACLE DBA Recommendations	2-15
	Preparing to Install ORACLE7 Server	2-21
	Preparing to Install Oracle Precompilers	2-25

	Preparing to Install SQL*Net Asynchronous	2-27
	Preparing to Install SQL*Net TCP/IP	2-28
	Preparing to Install SQL*Net V2	2-29
	Preparing to Install an Interchange	2-30
	SQL*Net DECnet	2-32
	SQL*Net TCP/IP	2-33
	Preparing to Install Office Automation Products	2-34
CHAPTER 3	INSTALLING ORACLE PRODUCTS	3-1
	Installer Overview	3-2
	How to Use the Installer	3-5
	The Install Actions Menu	3-13
	Mounting a CD-ROM and Starting the Installer	3-15
	Oracle Core Products Installation Messages	3-18
	SQL*Net V2 Installation Messages	3-20
	Oracle Office Automation Installation Messages	3-21
	Completing the Installation	3-22
CHAPTER 4	UPGRADING AND MIGRATING YOUR ORACLE SYSTEM	4-1
	Upgrade and Migration Paths	4-2
	Deinstalling Old Distributions	4-4
	Migrating from ORACLE V6 to ORACLE7	4-8
	Upgrading the ORACLE7 Server	4-11
	Import and Export	4-13
	Performing an Export Upgrade	4-15
CHAPTER 5	CONFIGURING YOUR ORACLE SYSTEM	5-1
	Recommendations for Configuring Your Database	5-2
	Options for Customizing your Installation	5-27
	Relinking	5-30
	Installing ORACLE in a Network Configuration	5-31

PART 2: ADMINISTRATION

CHAPTER 6	ORACLE SYSTEM ADMINISTRATION ON SUNOS	6-1
	Shared Memory Parameter Adjustments	6-2
	Relocating the SGA	6-3
	Running ORACLE in a Networked Configuration	6-5
	Performance Tuning	6-6
	Configuring Disk I/O for Better Performance	6-12
	Asynchronous I/O and Multiple DB Writers	6-13
	Raw Devices	6-14
	Buffer Cache Size	6-23
	Resource Limits and Disk Quotas	6-25

PART 3: PRODUCTS

CHAPTER 7	ORACLE CORE PRODUCT UPDATES	7-1
	SQL*Plus	7-2
	SQL*Forms V3.0	7-3
	SQL*Menu V5.0	7-5
	SQL*ReportWriter V1.1	7-6
CHAPTER 8	ORACLE PRECOMPILERS PRODUCT UPDATES	8-1
	Pro*Ada	8-2
	Pro*C	8-3
	Pro*COBOL	8-4
	Pro*FORTRAN	8-6
CHAPTER 9	SQL*NET V1 PRODUCT UPDATES	9-1
	SQL*Net V1 Overview	9-2
CHAPTER 10	SQL*NET V2 PRODUCT UPDATES	10-1
	Installing SQL*Net V2	10-2

	SPX/IPX Adapter	10 – 3
CHAPTER 11	OFFICE AUTOMATION PRODUCT UPDATES	11 – 1
	Oracle*Mail	11 – 2
CHAPTER 12	ORACLE ADDITIONS PRODUCT UPDATES	12 – 1
	Oracle XA Library	12 – 2
APPENDIX A	UNIX SUPPLEMENT TO ORACLE7 SERVER DOCUMENTATION	A – 1
	Administrator's Guide	A – 3
	Concepts Guide	A – 13
	Applications Developer's Guide	A – 15
	Migration Guide	A – 16
	Utilities Guide	A – 17
	SQL*Language Reference Manual	A – 19



FIGURES

FIGURE 2-1	Example of recommended SunOS directory file tree	2 - 6
FIGURE 2-2	Recommended \$ORACLE_BASE file tree	2 - 16
FIGURE 2-3	Installation Summary Sheet for the Oracle Base Structure ..	2 - 21
FIGURE 2-4	Installation Summary Sheet	2 - 22
FIGURE 3-1	Installer Task Overview	3 - 3
FIGURE 3-2	Function Key Mappings	3 - 7
FIGURE 3-3	Installer Sample Input Screen	3 - 8
FIGURE 3-4	Installer Online Help	3 - 9
FIGURE 3-5	Install Actions screen	3 - 10
FIGURE 3-6	Installer Product Installation Screen	3 - 11
FIGURE 3-7	Decision chart for Installer Actions	3 - 14
FIGURE 5-1	Recommended placement of ORACLE data files	5 - 12
FIGURE 6-1	Logical volumes	6 - 18

Figures – ii



PREFACE

This *Installation & Configuration Guide* is the SunOS-specific component of a set of manuals documenting installation, configuration, maintenance, and use of the ORACLE7 Server with related products. Information in this *Installation & Configuration Guide (ICG)* includes:

- procedure for installing the ORACLE7 Server (the first time, and subsequently)
- migration checklist for conversion to the ORACLE7 Server from an earlier version of ORACLE
- suggestions for assisting ORACLE7 Server users
- procedure for linking user programs that interface to the ORACLE7 Server

General information about the ORACLE7 Server for *all* operating systems is contained in the following publications:

- *ORACLE7 Server Concepts Manual*
- *ORACLE7 Server Utilities User's Guide*
- *ORACLE7 Server Administrator's Guide*
- *ORACLE7 Server SQL Language Reference Manual*
- *ORACLE7 Server Messages and Codes Manual*
- *ORACLE7 Server Migration Guide*
- *ORACLE7 Server Application Developer's Guide*
- *ORACLE7 Parallel Server Administration Guide*

Note: All references made throughout this book to specific chapters refer to chapters in this *Installation & Configuration Guide (ICG)*, except where noted.

Audience

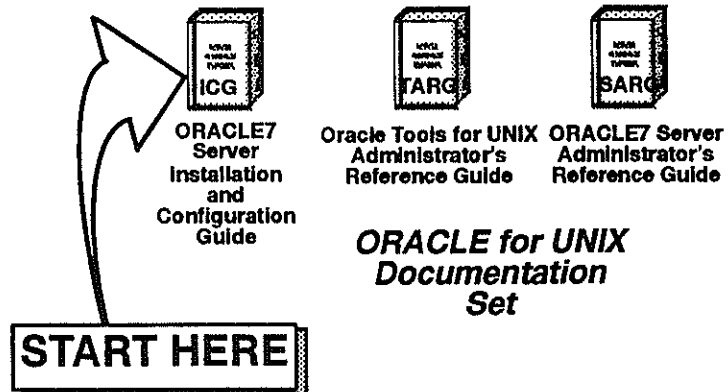
This manual is for people who install and maintain Oracle products on computers running the SunOS operating system. These people may also be assigned as ORACLE database administrators (DBAs) for the site.

The reader is assumed to have fundamental knowledge of the operating system under which the ORACLE7 Server will be running. No attempt is made to document features of the local operating system, except as they affect or are affected by the ORACLE7 Server.

How to Use this Manual

Before you attempt the procedures in this *Installation & Configuration Guide* you should review the *ORACLE7 Server Concepts Manual*, which explains the fundamentals of the ORACLE7 Server.

The figure below illustrates the UNIX-specific part of the documentation set you received with your Oracle products. It is intended to be used as a roadmap to the information you need to install, upgrade, or configure your Oracle products.



If you are:

Installing for the first time or upgrading:

Go to the "Installation Checklist" at the beginning of Part 1.



If you are:

Customizing Oracle products:

Go to the *Oracle Tools for UNIX Administrator's Reference Guide* for further information.



Oracle Tools for UNIX Administrator's Reference Guide



The *ORACLE7 Server for UNIX Administrator's Reference Guide* provides information for building, backing up, maintaining, administering, and recovering a database. Along with the *Installation & Configuration Guide* and *System Release Bulletin* (if available), refer to the *ORACLE7 Server for UNIX Administrator's Reference Guide* for information on:

- ORACLE system architecture under UNIX
- instructions for administering an ORACLE7 Server database on UNIX
- planning and running a complex database
- monitoring and tuning your system for optimum performance
- backing up and recovering the data in an ORACLE database
- UNIX command reference
- system error messages

The information formerly presented in the *ORACLE for UNIX Systems Error Messages Manual* has been incorporated in the *ORACLE7 Server for UNIX Administrator's Reference Guide*, and the *Error Messages Manual* has been eliminated.

Chapter 2 of the *ORACLE7 Server for UNIX Administrator's Reference Guide* provides a complete discussion of ORACLE system administration, including the following topics:

- environment variables
- special accounts and groups
- security considerations
- semaphores
- the SQL*DBA **monitor** command
- recovery tools
- trace files and alert files



The *Oracle Tools for UNIX Administrator's Reference Guide (TARG)* provides information for running Oracle products on UNIX-based platforms. The TARG is supplemented by information in this ICG. In particular, SunOS specific requirements for running products on your system are contained in Chapter 1 of this ICG; installation tasks and messages are contained in Chapters 2 and 3; and other SunOS-specific information is contained in Part 3 of this ICG. In addition, Oracle publishes a comprehensive set of documentation for all of its products. You will automatically receive all of the publications related to the products you purchase. For a complete list of Oracle publications contact your customer service representative.

Document Conventions

Conventions used in this manual differ somewhat from those used in other ORACLE documentation, including the generic ORACLE7 Server publications listed earlier. Since SunOS is case-sensitive, commands and filenames are not highlighted here by the use of upper-case letters. Instead, they are printed in boldface type. The case used in examples in this manual reflects precisely the way information should be entered by the user. The following conventions are observed in this manual:

Bold This type style is used for filenames, as in **init.ora**, and commands. Portions of the filename that may vary appear in italics, as in **sgadefx.dbf**.

Reserved words also appear in boldface in the text, to indicate that they are to be entered as is, and that they have reserved meanings within the ORACLE RDBMS. Use upper or lower case, exactly as shown.

ALL CAPITALS This type style is used for key names and are enclosed in square brackets, as in [RETURN].

Command Syntax

monospace Monospace is used for text that must be entered exactly as shown:

```
set echo off
```

italics Italics is used for variables in the text and for emphasis. In examples, variables are shown in italics. Substitute an appropriate value.

```
arg1
```

Vertical lines | Vertical lines are for alternative choices. The set of alternative choices is enclosed by curly braces if one

of the items is required, or by square brackets if the item is an optional alternative.

Curly braces {} Curly braces are for required items. Users must choose one of the alternatives.

```
.DEFINE { macro1 | macro2 }
```

Brackets [] Square brackets are for optional items.

```
cvtcert termname [outfile]
```

Ellipsis ... The use of ellipsis is for an arbitrary number of similar items.

```
CHKVAL fieldname value1 value2 ... valueN
```

The following symbols should always be entered as they appear in the command format:

period	.
comma	,
hyphen	-
semicolon	;
colon	:
equal sign	=
backslash	\
single quote	'
double quote	"
parentheses	()
accent grave	`

Other conventions

Note that "ORACLE7 Server" or "ORACLE" refer to the relational database server product from Oracle Corporation. "Oracle" refers to the corporation itself. The term "oracle" refers to an executable or account by that name.

Unless otherwise stated, examples use the Bourne shell (sh(1)) syntax.

Your Comments Are Welcome

We value your comments as a user of the ORACLE7 Server. As we write, revise, and evaluate, your opinions are the most important input we receive. Please use the Reader's Comment Form at the back of this manual to tell us what you like and dislike about this manual. Alternatively, you may contact us at the following address:

Jane Sokolow
UNIX Technical Publications Manager

Oracle Corporation
500 Oracle Parkway
Box 659407
Redwood Shores, California 94065
(415) 506-2456

Customer Support

Our Customer Support Services technical support answer line can be reached 24 hours a day. If, after following the instructions in the documentation, you need further assistance, please call:

(415) 506-1500

You will be asked a series of questions to help navigate you to the correct Oracle product support group. Be prepared to supply the following information:

- your CSI number (this helps us track problems recorded for each customer)
- the version numbers of the ORACLE Server and associated products
- the operating system name and version number
- details of error numbers and descriptions (write down the exact errors—it will help us track down the problem more quickly)
- a description of the problem



PART 1: INSTALLATION

Installation Checklist

The checklist below summarizes the major steps involved in installing and configuring ORACLE and its related products.

INSTALLATION CHECKLIST		
Step		Where to find
<i>Before Installation</i>		
☐ 1.	Check your Hardware and Operating System Software requirements.	in Chapter 1: <i>System Requirements for ORACLE</i>
☐ 2.	Calculate the total disk space you will need for all the products you wish to install.	in Chapter 1: <i>Disk Space and Memory Requirements</i> calculation tables
☐ 3.	Check any product-specific requirements.	in Chapter 1: <i>Oracle Tools Requirements</i> and/or in Chapter 2: <i>Preparing to Install</i> sections
<i>Installation</i>		
☐ 1.	Prepare your system environment.	in Chapter 2: <i>Where to Begin</i>
☐ 2.	Load the Oracle Installer.	in Chapter 3: <i>Starting the Oracle Installer</i>
☐ 3.	Run the installer and select your product(s) to install or upgrade.	in Chapter 3: <i>Using the Oracle Installer</i>
☐ 4.	Reply to the prompts and messages on screen.	in Chapter 3: <i>Oracle Tools Installation Messages</i>
<i>After Installation</i>		
☐ 1.	Complete any post-installation activities.	in Chapter 3: <i>Completing the Installation</i>
☐ 2.	Complete any ORACLE customizations.	in Chapter 3: <i>Completing the Installation</i>
☐ 3.	Configure your ORACLE system.	in Chapter 4: <i>Configuring your ORACLE System</i>

CHAPTER

1

KEY TO ORACLE ON SunOS

This chapter discusses SunOS-specific issues, including system hardware and software requirements for installing ORACLE.

The topics covered in this chapter are:

- system requirements
- product requirements
- disk space and memory requirements
- shared memory
- SunOS-specific issues
- SunOS-specific restrictions
- generic UNIX restrictions

System Requirements for ORACLE on SunOS

This section lists the hardware and software requirements for installing ORACLE7 Server on SunOS. The section following provides requirements for installing individual Oracle products on SunOS.

Hardware Requirements

Hardware Item	Requirement
CPU	A Sun SPARC-based system is required.
Memory	Please refer to the disk space and memory requirements charts in the next section.
CD-ROM Device	A CD-ROM drive that can read ISO 9660 standard format or High Sierra standard format.
Terminal	Supported terminals are listed in the READMEUNIX.doc file.
Controller	

Software Requirements

Software Item	Requirement
Operating System	SunOS Release 4.1.2/4.1.3 is required.

Compiler Requirements

Software Item	Requirement
Pro*Ada	Pro*Ada requires Sun Ada Compiler Version 1.1
Pro*C	Pro*C requires /usr/5bin/cc
Pro*COBOL	Pro*COBOL requires Sun COBOL Compiler Version 1.0
Pro*FORTRAN	Pro*FORTRAN requires Sun FORTRAN Compiler Version 1.4

Network Requirements

Software Item	Requirement
DECnet	SQL*Net DECnet requires SunLink DNI 7.0
SPX/IPX	SQL*Net SPX/IPX requires NetWare SunLink Version 1.0

Oracle Tools Requirements for ORACLE on SunOS

The following pages provide a separate table detailing installation requirements for the Oracle tools available for your operating system. The tables are grouped into the following categories:

- Core Products
- Precompilers
- SQL*Net V1
- SQL*Net V2
- Office Automation
- Additions

Core Product Requirements

SQL*Forms V3.0		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	running
Oracle*Terminal	1.0	installed
PL/SQL	1.0.36	installed
Third Party Software	Version	State During Installation
Sun C Compiler (required for C and non-C user exits)	/usr/5bin/cc	installed
Hardware Requirements		
none		

SQL*Menu V5.0		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	running
Oracle*Terminal	1.0	installed
SQL*Forms	3.0	installed
Hardware Requirements		
none		

Oracle*Terminal		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
ORAKIT	1.0	installed
Hardware Requirements		
none		

SQL*Plus		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Help Tables	n/a	installed
without installing Help Tables	n/a	n/a
Other Requirements		
Version 3.1 of SQL*Plus is required for ORACLE7 Server. SQL*Plus 3.0 will fail if used to access data from an ORACLE7 database.		
Hardware Requirements		
none		

SQL*Report		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	running
Hardware Requirements		
none		

SQL*ReportWriter V1.1		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	running
Third Party Software	Version	State During Installation
Communicae from Active Ingredients, Inc. is optional, but recommended for VT100 emulation.	n/a	n/a
Hardware Requirements		
none		

Precompiler
Requirements

Pro*Ada		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Sun Ada Compiler	1.1	installed
Hardware Requirements		
none		

Pro*C		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Sun C Compiler	version supplied with your O/S	installed
Hardware Requirements		
none		

Pro*COBOL		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Sun COBOL	1.0	installed
Hardware Requirements		
none		

Pro*FORTRAN		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Sun FORTRAN	1.4	installed
Hardware Requirements		
none		

SQL*Net V1
Requirements

SQL*Net Asynchronous Driver		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Hardware Requirements		
Standard RS-232 Async ports are required.		
Other Requirements		
Create a user account (default name is <code>sqlnet</code>) with a password and home directory.		

SQL*Net DECnet Driver		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
SunLink DNI	7.0	installed
Hardware Requirements		
A DECnet network with at least one DECnet Phase IV routing node is required.		

SQL*Net TCP/IP Driver		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Sun TCP/IP Bundle with OS		installed
Hardware Requirements		
An ethernet connection is required.		

SQL*Net V2 Requirements

SQL*Net V2		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Vendor's protocol-specific software	as required	installed
Refer to the chapter for the appropriate Oracle Protocol Adapter for further information.		

MultiProtocol Interchange		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
SQL*Net V2	2.0.12	installed
Oracle Protocol Adapters (at least two)	2.0.12	installed
Refer to the chapter for the appropriate Oracle Protocol Adapter for further information.		
Other Requirements		
The vendor's protocol-specific software must be installed for each supported protocol. Refer to the Oracle Protocol Adapter chapters for further information.		

SQL*Net DECnet Adapter		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
SQL*Net	2.0.12	installed
SunLink DNI	7.0	installed
Hardware Requirements		
A DECnet network with at least one DECnet Phase IV routing node is required.		

SQL*Net SPX/IPX Adapter		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
SQL*Net	2.0.12	installed
NetWare SunLink	1.0	installed
Hardware Requirements		
An ethernet connection is required.		

SQL*Net TCP/IP Adapter		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
SQL*Net	2.0.12	installed
Sun TCP/IP	Bundled with OS	installed
Hardware Requirements		
An ethernet connection is required.		

Office Automation
Requirements

Oracle*Mail		
Software Requirements	Version	State During Installation
ORACLE7 Server	7.0.12	n/a
Pro*C	1.5	installed
SQL*Plus	3.1	installed
Hardware Requirements		
An ethernet connection is required.		

Disk Space and Memory Requirements

This section provides the information required to calculate the disk space, database space, and memory requirements for your ORACLE7 Server distribution. The tables included, provide disk and memory requirements for each Oracle product available for your system. Follow the step-by-step procedure to calculate the disk storage and memory requirements for your specific installation. Note that these instructions provide generalized estimates of your disk and memory needs, rather than a precise calculation.

The Oracle products are divided into the following categories:

- applications development and standalone products
- extended functionality products
- CASE products
- precompiler products

Required Calculations

The calculations result in two values for total disk storage requirements and two values for system memory requirements. These values are defined as:

Distribution Space	is the disk space, in Megabytes, required to read in the product from the medium.
Database Space	is the database space, in Megabytes, the product requires in the database. Assess the number of ORACLE blocks required by dividing this amount by the default ORACLE block size for your machine.
First User Memory	is an estimate of the memory space, in Kilobytes, used when the first user runs the product. Refer to the discussion in each section for details of the exact meaning of this value for that category.
Memory Per Additional User	is an estimate of the memory space, in Kilobytes, used when each additional user runs the product. For exact meaning, consult each section's discussion.

Determining Available Disk Space

You can determine the free disk space in the partition in which you are going to install ORACLE by running the `df` command. Consult your operating system documentation for information on this command.

ORACLE7 and Extended Functionality Products

These products have no principal executables contained in their distribution. They consist primarily of software that is linked into other Oracle products to provide additional capabilities, such as networking, user interface, or procedural extensions to SQL.

Some of these, such as SQL*Net products, are linked into all of the principal executables of the standalone products. Others, such as PL/SQL, are only associated with certain applications development products, such as SQL*Forms V3.0, SQL*Menu V5.0, and the ORACLE kernel.

Per User runtime memory consumption for these products is difficult to measure, and in most cases (as with SQL*Net products, the distributed option, the procedural option, and the parallel server option) is insignificant compared to the standalone products.

To arrive at a basic estimate of your space needs for ORACLE7 and the Extended Functionality Products, perform these calculations:

1. **Select the products you want to install by placing a check mark in the first column of the table for each product.**
2. **Calculate the Distribution (Dist) space by subtotalling the values for the selected products in "A" at the bottom of the column.**
3. **Calculate the Database Space (DB Sp) by subtotalling the values for the selected products in "B" at the bottom of the column.**
4. **Transfer the First User Memory Space (#1 User) value for the ORACLE7 Server to the Subtotal "C" at the bottom of the column.**
5. **Calculate the Additional Users Memory Space.**

Calculate the Additional Users space by, entering the number of additional users you estimate for the ORACLE7 Server in the **Users** column.

Multiply **Users** by **Kb per** and place the result in **Total**. For example:

$$9 \text{ users} \times 186 \text{ Kb per} = 1674 \text{ K}$$

Transfer these results into Subtotal "D" at the bottom of the column.

6. **In the table's Summary section, enter the A, B, C, and D Subtotals.**

Add the #1 User Memory (C=) and Additional Users Memory (D=) and enter the result in Virtual Memory Total (E).

SPACE REQUIREMENTS ORACLE7 and Extended Functionality Products							
Disk Storage Requirements				Memory Space Requirements			
Use	Product	Dist. (Mb)	DB Sp (Mb)	#1 User (Kb)	Additional Users		
					Users	Kb per	Total
✓	ORACLE7 Server	11.8	12.5	52		x	=
SQL*Net V1:							
	SQL*Net Async	3.1					
	SQL*Net TCP/IP	3.1					
SQL*Net V2:							
	SQL*Net V2	3.1		992		72	
	TNS Listener	N/A					
	DECnet	3.1					
	SPX/IPX	3.1		952		64	
	TCP/IP	3.1					
	Interchange	3.1		1016		80	
	Interchng Data Pump	N/A		992		72	
	Configuration Tool	N/A		2134		198	
		↓ A ↓	↓ B ↓	↓ C ↓			↓ D ↓
	Subtotals →						
Summary							
A	Total Distribution Space	A=		(Mb)			
B	Total DataBase Space	B=		(Mb)			
	#1 User Memory	C=		(Kb)			
	Additional Users Memory	D=		(Kb)			
E	Virtual Memory Total	(C+D=)		(Kb)			

Applications Development and Standalone Products

Runtime memory consumption of the following products depends on one or two most frequently used executables. Some, such as SQL*Forms V3.0, differ between applications development and applications execution. For example, the development and test cycle of a given project may require three or four developers designing applications using the `sqlforms30` executable. Then, during the production cycle, more memory may be required if 50 users are expected to execute the finished applications using the `runform30` executable.

Keep in mind that First User memory consumption is extremely difficult to predict, since it depends heavily on the application. Furthermore, the number of cursors opened by applications can significantly affect the size of the associated shadow processes. The numbers given here are to be understood as the bare minimum.

1. Check the first column of the products you want to install.
2. Calculate the Distribution (Dist) space by subtotalling the values for the selected products in "A" at the bottom of the column.
3. Calculate the Database Space (DB Sp) by subtotalling the values for the selected products in "B" at the bottom of the column.
4. Calculate the First User Memory Space (#1 User) by subtotalling the values for the selected products in "C" at the bottom of the column.
5. Calculate the Additional Users Memory Space.

Enter the number of additional users you estimate for the product in the **Users** column and multiply by **Kb per** to determine the **Total**.

Subtotal these results and enter into Subtotal "D."

6. In the table's Summary section, enter the A, B, C, and D Subtotals.

Add the #1 User Memory (C=) and Additional Users Memory (D=) and enter the result in Virtual Memory Total (E).

SPACE REQUIREMENTS							
Applications Development and Standalone Products							
Disk Storage Requirements				Memory Space Requirements			
Use	Product	Dist. (Mb)	DB Sp (Mb)	#1 User (Kb)	Additional Users		
					Users	Kb per	Total
	SQL*Forms V3.0	0.1					
	Development (Char Mode)					× 345	=
	Runtime (Char Mode)					× 247	=
	SQL*Menu V5.0	0.1					
	Development (Char Mode)					× 319	=
	Runtime (Char Mode)					× 251	=
	SQL*ReportWriter	15.7					
	Development			1979		× 195	=
	Runtime			1723		× 171	=
	SQL*Plus	0.9		1437		× 117	=
	Oracle*Terminal	3.0		860		× 92	=
	SQL*Report	0.9					
	rpt			1160		× 72	=
	rpf			969		× 65	=
	Oracle*Mail	40.5		1966		× 206	=
		↓ A ↓	↓ B ↓	↓ C ↓			↓ D ↓
	Subtotals →						
Summary							
A	Total Distribution Space	A=		(Mb)			
B	Total DataBase Space	B=		(Mb)			
	#1 User Memory	C=		(Kb)			
	Additional Users Memory	D=		(Kb)			
E	Virtual Memory Total	(C+D=)		(Kb)			

Precompiler Products

Calculating memory usage of these products also requires differentiation of applications development and production cycles.

During applications development, a few programmers will be using the pcc executable extensively. During the production cycle, many more users may be executing the finished programs, which have been linked with the SQLLIB runtime library. Remember that each precompiler application spawns an oracle shadow process, so the same rule for including the Per Additional User memory consumption of the ORACLE kernel that applied to the applications development and standalone products applies to precompilers, as well.

Some languages, such as COBOL, involve compilers that do not generate native machine code. Their interpreters must be linked with the ORACLE runtime library, and their per user memory requirements can be much greater. Such estimates are beyond the scope of this document.

Disk space for the precompilers is somewhat redundant. It is calculated assuming only one language is installed on your system. When a second language is installed, the additional free space required is not as great as listed, since different languages share common libraries.

Development cycle calculations

First calculate the requirements for the development cycle:

1. Check the first column of the products you want to install.
2. Calculate the Distribution (Dist) space by subtotalling the values for the selected products in "A" at the bottom of the column.
3. Calculate the Database Space (DB Sp) by subtotalling the values for the selected products in "B" at the bottom of the column.
4. Calculate the First User Memory Space (#1 User) by subtotalling the values for the selected products in "C" at the bottom of the column.

5. Calculate the Additional Users Memory Space.

Enter the number of additional users you estimate for the product in the Users column and multiply by Kb per to determine the Total.

Subtotal these results and enter into Subtotal "D."

6. In the table's Summary section, enter the A, B, C, and D Subtotals.

Add the #1 User Memory (C=) and Additional Users Memory (D=) and enter the result in Virtual Memory Total (E).

SPACE REQUIREMENTS Precompiler Products: Development Cycle							
Disk Storage Requirements				Memory Space Requirements			
Use	Product	Dist. (Mb)	DB Sp (Mb)	#1 User (Kb)	Additional Users		
					Users	Kb per	Total
	PRO*Ada	1.1		2282		× 386	=
	Pro*C	1.1		2300		× 388	=
	Pro*COBOL	1.1		2301		× 389	=
	Pro*FORTRAN	1.1		2297		× 385	=
		↓ A ↓	↓ B ↓	↓ C ↓			↓ D ↓
	Subtotals →						
Summary							
A	Total Distribution Space	A=		(Mb)			
B	Total DataBase Space	B=		(Mb)			
	#1 User Memory	C=		(Kb)			
	Additional Users Memory	D=		(Kb)			
E	Virtual Memory Total	(C+D=)		(Kb)			

Use cycle calculations

To calculate the minimum memory requirements for a finished user program, perform the following steps:

1. Determine the text and data size of each executable, using the size command.
2. Add the text and data total to determine the #1 User memory requirement for the program (F).

$$\text{text} + \text{data} = \text{1st\_User\_Total}$$

3. Multiply the data value from Step 1 by the number of additional users of the program to determine the memory requirements for the remaining users of the program (G).

$$(\text{data}) \times (\text{Total\_Users} - 1) = \text{Addl\_User\_Total}$$

4. Multiply total number of users of the program by the Additional User Memory Space Requirement *given for the ORACLE kernel* (from the ORACLE7 and Extended Functionality Products Table) to determine total user kernel space (H).

$$(\text{Total\_Users}) \times (\text{ORACLE7 kernel}) = \text{User\_Kernel\_Space}$$

Add the results from Steps 2 – 4 for the total space requirement for the program. Enter the values for each program in the **Totals** column of the table below, and add the column to produce the **Total Memory Req (E)**.

MEMORY SPACE REQUIREMENTS Precompiler Products: Use Cycle				
Executable Program	#1 User Text + Data (F)	Addl Users Data × (users-1) (G)	Users × kernel (H)	Totals
		+	+	=
		+	+	=
		+	+	=
		+	+	=
		+	+	=
		+	+	=
		+	+	=
		+	+	=
				↓ E ↓
Total Memory Req (E)				

Final Calculations

Enter the A, B, and E totals from the individual tables into the table below. Total the three columns (including the additional Disk Space requirement) to determine the total **Distribution Space, Database Space, and Virtual Memory Space** you will need.

Total SPACE Requirements			
Transfer Tables	Dist Space (A)	DataBase Space (B)	Virtual Memory (E)
ORACLE7 and Extended Functionality Products			
Applications Development and Standalone Products			
Precompiler Products: Development Cycle			
Precompiler Products: Use Cycle	n/a	n/a	
Additional installation space for files generated during installation. See note 1.	+	n/a	n/a
Totals			

Note 1: This value includes space for database, log, listing files, library initialization routines, and space to install CASE*Dictionary.

Shared Memory

The shared memory feature of the SunOS IPC facility is required by the ORACLE system. The System Global Area (SGA) resides in shared memory; therefore, shared memory must be available to each ORACLE process to address the entire SGA.

To find the size of the SGA for a running database, enter the SunOS command **ipcs -b**. The SGA size, in bytes, is displayed under the heading "Shared Memory."

For more information on SunOS shared memory operations, refer to **ipcs**, **ipcrm**, **shmget**, **shmctl**, and **shmop** in the *Commands Reference Manual*.

See also the *System V Enhancements Overview* from Sun Microsystems, Inc.

Shared Memory Parameters

You can configure your system to allow shared memory by adding the line:

```
options IPCSHMEM # System V IPC
```

to the file `/usr/share/sys/sun4/conf/system_configuration_name`, and rebuilding the kernel.

Size Limits of the SGA

ORACLE uses shared memory segments for the SGA. The size of the SGA is limited by the address space of the CPU, available memory, and performance considerations.

For more information on SunOS shared memory operations, refer to **ipcs(1)**, **ipcrm(1)**, **shmget(2)**, **shmctl(2)**, and **shmop(2)** in the *Commands Reference Manual*.

SunOS-Specific Issues

Key to O/S Documentation

Operating system documentation available from Sun Microsystems, Inc. includes the following important manuals:

- *System Administration for the Sun Workstation*
- *Commands Reference Manual*
- *System V Enhancements Overview*

ORACLE man Pages

Note: Online manual pages are not available with the V7.0.12 release of ORACLE RDBMS, but will be in future.

The automated installation procedure described in the "Installation Tutorial" chapter prompts you to install them on your system.

If you decide to install them, they will be added to the system **man** page directory, **/usr/man/cat1**.

You can now access them as you would any SunOS manual page. For example, the following command displays the manual page that documents the **orasrv** utility.

```
$ man orasrv
```

SunOS Shell

The shell is the most frequently used utility program on a standard SunOS system. It is the SunOS system's command interpreter, prompting the user for commands and causing the operating system to obey them.

In this guide, Bourne shell (**sh**) conventions are used in the examples. For instance, a dollar sign (\$) is used to represent the prompt. This is standard for the Bourne shell. The C shell generally uses a percent sign (%) as prompt.

On SunOS systems you can use either the Bourne shell (**sh**) or the C shell (**csh**) with the ORACLE RDBMS.

SQL*DBA defaults to the Bourne shell if your **SHELL** environment variable is not set. Note that you can invoke the shell from SQL*DBA by entering an exclamation point (!).

Block Size and File Size

Although data storage space is often measured in megabytes (one Mb = 1024 * 1024 bytes), the SunOS operating system and the ORACLE RDBMS each perform input and output in units of data storage called "blocks." Note that the size of the operating system blocks are not necessarily equal to ORACLE blocks.

In Release 7, ORACLE Server block size is not fixed. You can set its value when creating a database by changing the **DB_BLOCK_SIZE** parameter in the **init.ora** file. Refer to the "Installation Options" chapter for a description of **init.ora**.

Changing the ORACLE block size may change ORACLE performance, depending on your disk hardware, file system, and application. You may wish to experiment with different block sizes to see what is most efficient for your configuration. The size of 2K bytes should be adequate for all circumstances, but some benefits may be gained by optimization.

Specifying ORACLE block size

On your system, ORACLE has a default block size of 2K and a maximum block size of 8K.

You can set the actual block size to any multiple of 2K between 2K and the maximum block size, inclusive.

The optimal block size on your system is typically 2K, but varies with your applications. To create a database with an ORACLE block size that is different from the default, specify the new block size in the **init.ora** file by adding the line:

```
db_block_size=new_block_size
```

You may wish to experiment with block size before transferring your data to a new database, since database creation is required for a block size change.

To specify the database block size during the installation, modify the **init.ora** file in **\$ORACLE_HOME/dbs** after reading the tape, but before starting the installation. This affects only the default database.

Maximum number of extents

The first block of each segment contains an extent map for each extent. The maximum number of extents allowed is therefore a function of the database block size and the size of each extent map entry.

The only way to increase the limit is to change the database block size (which requires a full export/import). The table below shows the database block size relative to the greatest **maxextents** value.

Database Blocksize vs maxextents	
Database Block Size	Greatest maxextents Value
512 bytes	25
1K	57
2K	121
4K	249
8K	505

Specifying database and log file size

The database file should be big enough to hold the data dictionary, all user data, plus a margin for overhead and indexes. The recommended minimum size for a database file is 5 Mb. You must have a minimum of two log files, which can be relatively small (perhaps 100 K each). If you have multiple database files, then the first file need only be big enough to hold the data dictionary and the initial rollback segment. Refer to the *ORACLE RDBMS Database Administrator's Guide* for detailed information on this topic.

The automated install script uses the database and log file sizes given by the corresponding environment variables. If ORACLE has already been installed, you can specify database and log file size when you use SQL*DBA to execute the SQL statement **create database**. Here is an example:

```
create database Test
logfile '?/dbs/log1.f' size 500K, '?/dbs/log2.f' size 500K
datafile '?/dbs/db1.f' size 5000K;
```

Filename

On your system, the full path names of certain important files and directories are shown in the following table:

Directory Locations	
Directory	Use
/usr/local/bin	local bin directory
/usr/oracle	ORACLE home directory
/usr/local/bin	oraenv directory

Filename restrictions

As of ORACLE V6.0.27, the format of some filenames, including all SQL scripts under `$ORACLE_HOME/rdbms/admin`, has changed to limit the length to a maximum of 14 characters; this restriction is encountered on some systems. For those systems, the `ORACLE_SID` is restricted to four characters or less. This restriction does not apply to SunOS.

Environment Variables

On your system, the default values for the basic environment variables are shown in the following table:

Environmental Variables	
Environmental Variable	Value
<code>DBS_SIZE</code>	10 Mb
<code>ORACLE_HOME</code>	<code>/usr/oracle</code>
<code>ORACLE_PATH</code>	<code>/usr/oracle/bin</code>
<code>ORACLE_LPPROG</code>	<code>/usr/ucb/lpr</code>
<code>ORACLE_LPARGS</code>	<code>-p</code>
<code>ORACLE_LPSTAT</code>	<code>/usr/ucb/lpq</code>
<code>ORACLE_PAGER</code>	<code>/usr/ucb/more</code>
<code>ORACLE_SID</code>	Name of ORACLE database owner.
<code>SHELL</code>	<code>/bin/sh</code>
<code>TWO_TASK</code>	<code>P:</code>
<code>APIPATH</code>	<code>\$ORACLE_HOME/rdbms/admin</code>

Upgrading Your System

When upgrading your system from earlier versions of ORACLE RDBMS, you must either perform an export upgrade or use the migration utility. For more information on export upgrades, see the "Installation Options" chapter. The migration utility is described in the *ORACLE RDBMS Version 6 to Version 7 Migration Guide*.

export upgrade when you are upgrading from V6.0.24 or earlier, or when upgrading from V7.0.7

migration utility when you are upgrading from V6.0.26 or later versions of V6.0

Note: If your current version of ORACLE RDBMS is V5.1, migrate first to V6 before migrating to V7. Refer to the V6 documentation for instructions on migrating from V5.1 to V6.

Terminal Characteristics

Using the @ character

This section describes SunOS conventions relating to terminals.

SQL*Plus uses the *@filename* convention to indicate an indirect command file. In some SunOS systems the @ character is chosen as the default line kill character, which effectively erases an entire line of input.

To avoid unexpected results, we recommend that you redefine the @ character using the **stty** command. Refer to **stty(1)** in the *Commands Reference Manual* for more information on changing terminal characteristics.

Alternatively, you can use a backslash (\) to escape the @ character, either on the command line or within shell scripts. For example,

```
$ sqlplus \@demo
```

Resetting the terminal

SQL*Forms uses raw input mode to capture all data typed at the keyboard. If SQL*Forms terminates abnormally, it can leave the terminal in raw input mode, and no input will be echoed. This has the appearance of "hanging" the terminal. To free it, enter the command:

```
$ reset ^j
```

where ^j indicates [CTRL]j, which is the same as [LINEFEED]. This will reset the terminal to a usable state.

See **reset(1)** in the *Commands Reference Manual* for more information.

SunOS-Specific Restrictions

orasrv Location	On SunOS hosts, orasrv is typically started from <code>/etc/rc</code> .
Housekeeping Messages at Install	You may disregard certain housekeeping messages related to the ranlib process, which will appear in the install.log file upon installation of ORACLE. These messages read: <pre>ranlib: warning: library (module): no symbol table</pre>
Concurrent Process Limit	The number of concurrent ORACLE processes (init.ora parameter processes) for each ORACLE system is limited by the SunOS-imposed limit for the maximum number of allowed semaphore identifiers system-wide.
Swap Space Restriction	Very large SGA's can cause the system to run out of swap space. Note in particular that given a 17 Mb swap partition, one ORACLE system will support a maximum of about eight concurrent users. With two ORACLE systems running, each will support about two concurrent users. Increasing the swap partition size, or using several partitions for swapping will increase the number of concurrent users possible.
Resizing Windows	Some ORACLE programs are not sensitive to window size. SQL*Plus defaults the LINESIZE variable to 80, and the PAGESIZE variable to 25. These variables are not automatically adjusted for different window size; Appendix B of the <i>SQL*Plus User's Guide</i> explains how to change them. SQL*Forms now uses the actual size of the shelltool window in which it is run. SQL*ReportWriter will not resize, and its default size is 24 lines by 80 columns.
Type 3 and 4 Keyboards	All key definitions released with ORACLE V7 products are designed for BOTH the Sun Type 3 and Sun Type 4 keyboards. However, if you have customized definitions for an application that specifically address the Type 3 keyboard, and you do not wish to convert them, you can make a mechanical change to the Type 4 keyboard to make it respond like a Type 3 keyboard.

To do this, find the plate of dipswitches under the upper right corner of the Type 4 keyboard. Move the first one (#1) to the ON position (the default is OFF).

Archive Monitor

Due to an operating system restriction, a SunOS #1 error occurs when you try to close the tape device in order to mount the next tape. To work around this inappropriate error, before implementing tape archival, have your System Administrator use **chown** to change the ownership on the tape device (e.g., **/dev/rst0**) to the owner of the **oracle** process. During tape archiving, the drive will be used exclusively by the **oracle** process.

Full Screen Tools

Any full screen tools that use .crt terminal screen drivers (such as SQL*Forms, SQL*DBA monitor, SQL*Menu, and SQL*Calc) work properly in both **shelltool** and **cmdtool**. In **shelltool**, the **TERM** environment variable must be set to **sun**; in **cmdtool**, it must be set to **sun-cmd**.

Other full screen tools, such as SQL*ReportWriter and Oracle*Mail, function properly only in **shelltool** windows.

If you are using one of the full screen utilities using **cmdtool**, and you exit abnormally, your cursor will be left in a **shelltool** state (a solid, unblinking block). To return the cursor to the **cmdtool** state (a blinking triangle), enter a terminal reset escape sequence:

```
$ echo [ESC][>4h
```

Note: The [ESC] key is represented on the screen by "^[". This means you will see "^[[>4h" on the screen if you enter the correct sequence.

Underlining in Oracle Applications

On the Sun-4, among the Oracle*Terminal resource files that you can choose to use (via the **-c** option) with character mode are:

sun.r	To be used in a commandtool or shelltool window, which do <i>not</i> support underlining. The Oracle*Toolkit underline attribute therefore has no effect. This affects any Oracle application that refers to this file (such as SQL*Forms V3.0, SQL*Menu V5.0, SQL*ReportWriter, CASE*Dictionary, etc.).
xtermsun.r	To be used in an xterm window. Underlining is supported.

SQL*Net Updates

Linking Problems and SQL*Net Drivers Libraries: If any sample *language* program does not link properly for the reason that symbols such as `osnttt`, `osndnt`, or `osnasy` are unresolved, this means one or more of the SQL*Net drivers have not been installed. Correct this problem either by installing the missing drivers, or put references to the SQL*Net driver libraries in the `prolanguage.mk` file. The references must go in the link line after the `libsqlnet.a` entry.

SQL*Net Asynchronous: Before installing SQL*Net Asynchronous, create a user account (the default account name is `sqlnet`), with a password and home directory.

SQL*Net DECnet: The DECnet Network Services Protocol is called `dninsp`, and the name of the SQL*Net DECnet user account should be `dni`.

Pro*COBOL

Pro*COBOL no longer assumes that the COBOL home directory to be `/usr/lib/cobol..`. Therefore, you have to set the `COBDIR` environment variable to the directory that your COBOL compiler resides in. See your COBOL compiler documentation for more information.

Restrictions and Work-arounds

Unsupported Products	The ORACLE man pages are not currently supported.
Install Comments	Every database must have a unique <i>sid</i>. When you run the Installer, your login ID must belong to the group that you designate during installation as the "DBA group" (that is, the group with DBA privileges). The default group-id for this group is dba. It is membership in this group that assigns access to the "specially privileged" SQL*DBA functions. If you install any SQL*Net network drivers other than the defaults, you must relink all product executables. When some products are installed, the Installer modifies the startup file (either .login or .profile) of the <i>oracle</i> user. For example, some products require special environment variables or modifications to existing ones. Oracle Installer puts these changes into the startup file. After you have installed the product, you must make the operating system read the altered startup file. You can do this either by logging out of the operating system and then logging in again, or by running source on the startup file (in C shell), or by executing the file (in Bourne shell).
Default Specification of Database Files	In SunOS, the default directory location for database files is the \$ORACLE_HOME/dbs directory. This applies when you are using the create tablespace command. If you do not give a full pathname for datafiles created with this command, the files will be placed in the above directory.
ops\$ login and SunOS	SunOS treats a dollar sign (\$) as the beginning of an environment variable. Therefore, when you specify an ops\$ login (a remote login) on the command line or in a script, first "escape" the \$ with a backslash (\). For example, user-id scott should specify ops\scott when logging in remotely. Automatic logins (ops\$ logins) are not allowed for the root user-id, because of security problems that can arise.
Encrypted Passwords	Because password encryption on ORACLE can differ based on your operating system, users migrating from one operating system to another need to be aware that they might have to re-grant the connect privilege to users.

Product Compatibility with ORACLE

Although some products in the ORACLE7 Server product set may have the same version number as in the ORACLE Version 6.0 product set, the install process is different. You must order the Release 7.0 product set for use with ORACLE Server.

transactions_per_rollback_segment

The `transactions_per_rollback_segment` parameter in the `init.ora` file can be increased, but is subject to the SunOS hard-coded limit of 61. You may encounter this limitation if you only have one rollback segment besides the SYSTEM one, and there are many running transactions requiring its use. Work around this limitation by creating multiple smaller rollback segments.

SunOS Restriction on Exit Codes

Errors will not be displayed correctly when the exit (or return) code is greater than 127.

maxdatafiles Limit

The chapter "Managing the Size and Files of the Database" in the *ORACLE7 Server Administrator's Guide* states that the absolute maximum number of database files is set with the `maxdatafiles` parameter to the `create database` statement. The ORACLE limit is 1500, but on your machine this is superseded by a system-specific limit.

The SunOS-specific limit for `maxdatafiles` is 1022

UNIX Syntax and Import Example

In the *ORACLE7 Server Utilities User's Guide*, the first example given in the section titled "Sample Import of Tables from One User to Another" uses an asterisk to specify all tables. For SunOS users, this line should read:

```
imp system/manager file=scott.dmp fromuser=scott  
touser=rosemary tables='(*)'
```

That is, the parentheses at the end of the line must be surrounded in single quotes.

Importing from VM to UNIX

If you are using the `import` utility to import data from a VM, MVS or VMS database, you must use the following syntax:

From VM, MVS or VMS enter:

```
$ EXP RECORDLENGTH=n username/password
```

On SunOS enter:

```
$ IMP RECORDLENGTH=n username/password
```

The variable "*n*" is the size of the records in the export files and must match exactly the size of the records in the import file for the transfer to work. There is no case sensitivity in this command, and no username prompting.

osh

The ORACLE shell **osh** is supplied but not mandatory. You may remove it if you wish.

Shutdown Errors

If you use the **immediate** or **abort** options of the **shutdown** command, errors **ORA-7265** and **ORA-1089** will appear in your **alert_sid.log** file. You can ignore these errors.

SQL*Plus

SQL*Plus version required for ORACLE7

Version 3.1 of SQL*Plus is required for ORACLE Server. Using SQL*Plus 3.0 will fail when used to access data from an ORACLE database.

copy command

The **copy** command in SQL*Plus is supported without restrictions on homogeneous systems. Homogeneous systems are defined as SunOS systems running the same version of SunOS.

The **copy** command may also work between heterogeneous machines. If it should fail between two heterogeneous machines, test the connection using **rcp** or **ftp**. Bugs in vendor-supplied networking software may prevent even **rcp** or **ftp** from functioning properly between heterogeneous systems. The same bugs may cause the **copy** command to fail. Note that **rlogin** is not a satisfactory test for connections because it does not send and receive large amounts of data.

If the **copy** command does not function between your heterogeneous systems, you can still work around this problem. Create a database link to the system and user-id which contain the table you wish to copy. Then log on to the system to which the table should be copied, and enter this SQL*Plus command:

```
create table newtable as (select * from  
table@database_link_name)
```

This will select all rows and columns from the original table on the remote system, and enter them in the new table, effectively copying the original table.

Spool Files

If a directory listed in **ORACLE_PATH** contains scripts that may be spooled, there is the possibility that spool output files may be written to that directory, overwriting the original scripts. To avoid problems, directories named in **ORACLE_PATH** should not contain spool files. Alternatively, spool files should be renamed so that they will not be overwritten.

Fast Two-Task Driver

When the Fast Two-Task driver is used and a utility fails abnormally, shared memory segments may not be deallocated. If another utility is executed, there may not be enough shared memory to attach and the second utility will fail.

To remedy this problem, you can use the **ipcs** and **ipcrm** utilities to manually deallocate the shared memory segments.

8-Bit Input

In ORACLE V6.0 releases earlier than 6.0.27, in order to enter 8-bit input (foreign languages) to SQL*Forms applications, you had to enter either **stty -istrip** or **stty pass8** before invoking SQL*Forms. These commands tell the terminal driver not to strip the eighth bit. As of V6.0.27, you no longer need to do this, because the terminal driver is set to accept 8-bit input automatically when you invoke SQL*Forms.

Note: Generally, the 8-bit mode will work only through hard-wired ports. The **rlogin** and **telnet** facilities will strip the eighth bit. A work-around for **rlogin** is to enter **rlogin -8**. The **-8** is a flag that turns 8-bit support on.

On BSD-based UNIX systems (or dual universe systems running in the **ucb** universe) problems can occur when you run a VT200 series terminal in VT100 emulation mode with 8-bit communications, and then enter an ORACLE utility that uses the ORACLE VT200 or VT220 **crt** definition.

The affected utilities may include SQL*Forms, SQL*Menu, and Easy*SQL. The VT2XX **crt** definitions for these utilities automatically switch your terminal from whatever control mode it was in initially, to "VT200 7-bit controls". This mode enables the **crt** definition to access the "F" keys at the top of the keyboard (which the VT100 **crt** definition does not use).

The problem arises if your terminal's communications mode is still set to 8 bits; to run in "VT200 7-bit controls" mode, the terminal's communications mode must also be set to 7 bits. Your screen will be corrupted if the communications mode is set to 8 bits. While ORACLE **crt** definitions do change the terminal's control mode, they do not change the terminal's communications mode.

This problem concerns only the machine you are directly logged on to, not the machine your application is running on. Here is an example in which "N" is a machine which does *not* interpret the terminal modes correctly, and "D" is a machine which *does* interpret them correctly,

- If you are logged on to N and you try to run SQL*Forms over the network on D, the screen will be corrupted.
- If, however, you are logged on to D and you run SQL*Forms over the network on N, your screen will be fine.

To avoid problems with VT2XX `crt` definitions, manually set the terminal to 7-bit communications mode whenever you invoke SQL*Forms.

Please note also that the VT2XX `crt` definitions reset the terminal to VT100 mode when you exit from ORACLE utilities. For example, if your terminal is running SQL*Forms in VT200 mode, the `crt` definition will change your terminal to VT100 mode when you exit from SQL*Forms. To avoid this, simply change the VT2XX `crt` definition. For details, please see the manual for your VT220 terminal.

Post Processing of Terminal I/O

Some programs turn off post processing of terminal I/O and do not turn it back on again when exiting.

To avoid problems caused by this situation, add the line `stty -nopost` to your `.login` or `.profile` file.

SHELL Variable

To ensure that you can use Escape to Host in SQL*ReportWriter, you must set the `SHELL` environment variable to `/bin/sh`. You can do this manually, or enter it in your profile file.

Corrections to the SQL*Forms V3.0 Designer's Tutorial

Corrections to the *SQL*Forms V3.0 Designer's Tutorial* (part no. 3302-V3.0, dated 0789) are listed by page number in the `tutorial.doc` file, which is located in the `$ORACLE_HOME/forms30/doc` directory.

Product Obsolescence

End of Life for Sun-3

The 6.0.33 release of the ORACLE RDBMS and Oracle products is the final formal release of new product versions for the Sun-3 platform. Oracle provided customer support for the Sun-3 platform through June, 1992.

This decision is based on Sun Microsystems' aggressive End of Life plan for the Sun-3 platform, including its decision not to release new systems software for Sun-3, and its aggressive SPARC (Sun-4) upgrade programs. Should you choose to upgrade to SPARC, please contact your Oracle sales representative to arrange for an upgrade to Sun-4 releases of Oracle products.

ODL

The ORACLE Data Loader has been superseded by SQL*Loader. Refer to the *ORACLE7 Server Utilities User's Guide* for more information on SQL*Loader.

To convert ODL control files to SQL*Loader control files, enter the following:

```
odl2ldr odl_control_file loader_control_file
```

where:

odl_control_file Specifies the ODL control file being converted.

loader_control_file Specifies the target SQL*Loader control file.

2

PREPARING TO INSTALL ORACLE PRODUCTS

This chapter is designed to guide you through recommended preinstallation activities for the SunOS system administrator and the ORACLE DBA, and to provide you with the necessary steps to plan your installation.

This chapter is divided into the following sections:

- where to begin
- introduction to the Oracle Base Structure
- UNIX system administrator recommendations
- ORACLE DBA recommendations
- Oracle product preinstallation activities

The recommendations in this chapter apply to single instance, single database configurations as well as to multi-instance, single database models (ORACLE Parallel Server).

Where to Begin

This section guides you to the documentation useful for your particular installation. You should be familiar with the installation requirements and issues covered in the “Key to ORACLE on SunOS” chapter before you read further. It is assumed that you have a working knowledge of your operating system and of the ORACLE7 Server.

You must decide where to begin your installation, based on the current status of previous ORACLE versions on your system.

If you are installing ORACLE7 Server for the first time:

- follow the instructions in this chapter

If you are migrating from Version 5.1:

- follow the instructions in the “Migrating from ORACLE V5.1 to V6.0” appendix in the *Version 6 Installation and User’s Guide* first, then use the
- *ORACLE7 Server Migration Guide*, with the
- “Migrating from ORACLE V6 to ORACLE7” section of the “Upgrading and Migrating Your ORACLE System” chapter

If you are migrating from Version 6.0.26 or earlier versions of 6.0:

- follow the instructions in the “Performing an Export Upgrade” section of the “Upgrading and Migrating Your ORACLE System” chapter

If you are migrating from Version 6.0.27 or later versions of 6.0:

- follow the instructions in the *ORACLE7 Server Migration Guide*, with the
- “Migrating from ORACLE V6 to ORACLE7” section of the “Upgrading and Migrating Your ORACLE System” chapter

If you are upgrading from ORACLE7 Server Release 7.0.9:

- follow the instructions in the “Deinstalling Old Distributions” section in that chapter to deinstall your 7.0.10 distribution, and then
- follow the instructions in the section “Upgrading the ORACLE7 Server: Upgrading with the Installer” in the “Upgrading and Migrating Your ORACLE System”

- follow the instructions in the "Performing an Export Upgrade" section of the "Upgrading and Migrating Your ORACLE System" chapter

If you are upgrading from ORACLE7 Server Release 7.0.10:

- follow the instructions in the "Deinstalling Old Distributions" section in the "Upgrading and Migrating Your ORACLE System" chapter to deinstall your 7.0.10 distribution
- follow the instructions in the "Upgrading ORACLE7 Server: Upgrading with the Installer" section of the "Upgrading and Migrating Your ORACLE System" chapter

If you are upgrading from ORACLE7 Server Release 7.0.11:

- follow the instructions in the "Upgrading ORACLE7 Server: Upgrading with the Installer" section of the "Upgrading and Migrating Your ORACLE System" chapter

Introduction to the Oracle Base Structure

The recommendations in the first two sections of this chapter are based on the "Optimal Flexible Architecture (OFA) Standards Document" written by Oracle Consulting and tested at hundreds of ORACLE sites. This set of recommendations, referred to as the Oracle Base Structure, are not mandatory, but provide a structure flexible enough to accommodate system growth coming from any of several directions:

- increasing data volume within an existing database
- acquisition of additional disk hardware for an existing system
- creation of a new database in an existing operating system environment

The Oracle Base Structure is especially recommended for large sites. For sites where ORACLE is already installed and a particular directory structure is already determined, the recommendations in the following sections may not be optimal. In either case, you should read through *all* of the recommendations first and determine if the Oracle Base Structure will benefit you. For Oracle Base Structure post-installation recommendations, see the "Configuring Your ORACLE System" chapter.

If you decide not to use the Oracle Base Structure, you may proceed directly to the "Preparing to Install ORACLE7 Server" section for necessary preinstallation activities for ORACLE7 Server and Oracle products.

The Oracle Base Structure balances your database files across all the disk drives on the system in a highly organized manner, making maintenance, growth, migrations, and administration easier and more efficient.

The Oracle Base Structure makes ORACLE maintenance easier by storing database files in a simple, consistent way. Files of one database are completely separated from files of another, so the risk of corrupting production data by accidentally "crossing" files is minimized.

Adherence to the recommended patterns enables the administrator to manipulate entire sets of data with a single UNIX command. This feature makes it possible to create UNIX programs that can accommodate any tablespace creation or growth, database creation, disk hardware addition, or disk load balancing file movement without modification.

SunOS System Administrator

This section lists exactly what the DBA needs from the **root** user or SunOS system administrator. It also addresses the administrator's concerns about security and resource consumption.

The following is a brief list of the recommended steps detailed in this section:

Steps for the SunOS file system:

1. Create mount point directories for ORACLE system data.
2. Create user home directories.

Steps for SunOS logins:

3. Create groups.
4. Create the ORACLE owner login.
5. Create the DBA logins.
6. Create an Applications owner login (only if using Oracle Applications).
7. Create ORACLE user logins.

Steps for ORACLE files and data:

8. Create ORACLE files that belong in SunOS directories.
9. Create ORACLE data directories.

Raw devices:

10. Consider using raw devices.

The SunOS File System

The file system described in the following section is a simple way to permit the database administrator to balance I/O across the entire system with absolutely no compromise to other users on the machine. The purpose of encouraging the organized use of small portions of several disk drives, is to increase the potential for optimal database performance while simultaneously ensuring a clean, flexible way to administer other kinds of files on the disk drives that house ORACLE data.

We recommend that ORACLE-related data and product files be distributed across different disk drives, as shown in the following figure. This helps balance I/O, and if a single disk should crash, your database is still available.

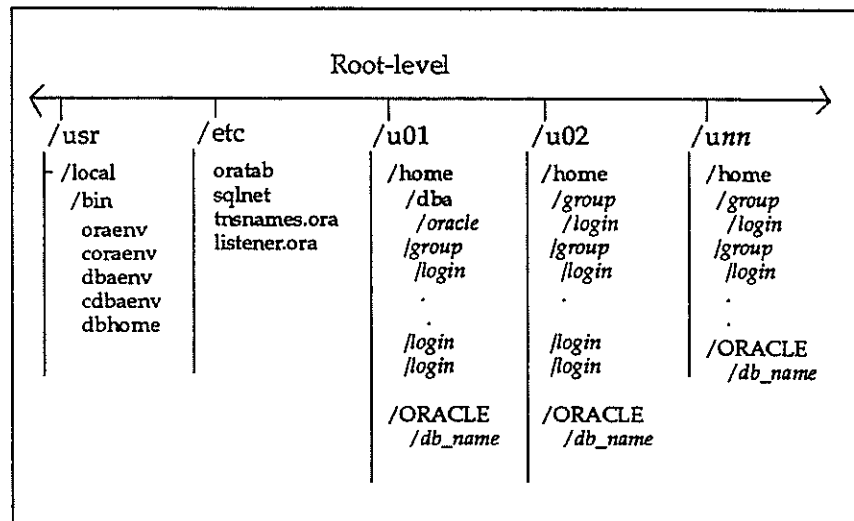


FIGURE 2-1 Example of recommended SunOS directory file tree

Create mount point directories

Name all mount point directories that can contain ORACLE system files using the pattern `/unn` or `/unnn`, where *n* is a digit in the set [0-9]. Give these directories owner **root**, group owner **root** or **wheel**, and mode 755.

The names of some root-level mount point directories are dictated by the UNIX operating system. The `/usr` and `/etc` directories are prime examples. Experienced SunOS administrators have learned to isolate certain critical directories into their own mount points in order to facilitate the administration of the machine.

The mount point directories containing ORACLE system data are given the names `/unn` solely to distinguish them from one another. The pattern `/unn` is a typographically efficient name that is reminiscent of "user data," the only type of categorization we are permitted to make at this level in the directory tree. Other acceptable patterns are `/dnn`, for disk data, etc.

Including a unique numeric string in the mount point directory name uniquely identifies the mount point. Using the `/unn` recommendation (name followed by a two-digit unique key) allows the use of a single pattern to match up to 100 mount point directory names (e.g., `/u00`

through /u99). Using /unnn (three-digit key) allows the use of up to 1,000 mount point names.

At key levels in the file system such as this one, a directory name such as **oracledata** would limit the contents of the directory. If, at a later time, you placed some non-ORACLE data in that directory for the purpose of space allocation, the pathname immediately becomes misleading.

Whether you decide to use a two- or three-digit key, use your selected pattern consistently across mount points and pad the numbers one through nine with a preceding zero so that the directory names will align visually and sort properly in the output produced by device reporting utilities. For example, no zero padding would result in the following sort order: /u1, /u10, /u11, /u2, /u3. Zero padding, on the other hand, would result in: /u01, /u02, /u03, /u10, /u11.

Mode 755 permits any user to enter and search the root-level directories, but not to create files there. Write privileges will be administered at a deeper level in the file tree.

Create user home directories

Create a directory named **home** in each SunOS mount point subtree, and name every non-**root** user home directory using the pattern **/home/login** or **/home/group/login**. Give these directories owner **root**, group owner **root**, and mode 755.

Placing a **home** directory in every mount point subtree maximizes flexibility in finding large quantities of disk space for system users, including owners of large quantities of applications software.

Mode 755 permits any user to enter and search the user data-level directories, but not to create files there. All write privileges will be administered at a deeper level in the file tree.

Using a consistent pattern ensures that all user home directories can be manipulated as a unit. User home directories are the most frequently modified directory trees on the system. This recommendation allows you to create zero-maintenance backup scripts to archive the set of system files that changes most frequently.

Administrators at larger sites may choose to use an additional level in user home directory names to separate users by function, using the pattern **/home/group/login**.

For example, using the group layer in the UNIX directory structure, a UNIX administrator would be able to back up the **dba** group member directories by issuing the following command:

```
$ find */home/dba/* -print | cpio -ov > /dev/rst0
```

SunOS Logins

This section gives recommendations for creating the following:

- groups
- ORACLE owner
- database administrators
- Oracle Applications owner
- ORACLE users

Create user groups

Some administrators may need to separate sets of SunOS logins into different groups which are defined in the `/etc/group` file. By using groups, you can limit file access to well-defined lists of users. The following is an example `/etc/group` file:

```
$ sort -nt: +2 -3 /etc/group
root:*:0:root,adm
other:*:1:daemon,sync,ttty,who,uucp,nuucp
bin:*:2:bin,daemon,lp,games
sys:*:3:bin,sys,adm
adm:*:4:adm,daemon
...
dba:*:101:rrudzki,cmillsap,cmazza
oaa:*:102:cshallah,jgrubel,tkemp
clerk:*:103:madler,bepstein,csherida,lwarneka,ljamtgaa
nobody:*:65534:nobody
```



For more information on groups see the “Administering ORACLE” chapter in the SARG.

Create the ORACLE owner login

To accomplish the setup of multiple database administrators, we suggest that you create a SunOS login named **oracle** (referred throughout this book as the *oracle* user) that will act solely as the owner of the ORACLE program files. The *oracle* user does not represent a person, but is considered a “responsibility” login. The only time anyone should log in as *oracle* is to perform ORACLE software maintenance, to install patches or perform upgrades. Day-to-day database administration is performed by *dba* group members using their personal logins.

Create a SunOS login with the following properties:

- login name *oracle* or, for sites with multiple DBA teams, use login names that connote the team names
- user identification (UID) number between 3 and 32767

- default group identification (GID) number between 2 and 32767, corresponding to the group named **dba** or, for sites with multiple DBA teams, use group names that connote the team names
- password (GCOS) field specify "ORACLE database owner" for the user name (append, **universe(att)** for systems with **universe(1)**)
- home directory matching the pattern **/*/home/login** or **/*/home/group/login**
- login shell **/bin/ksh** or **/bin/csh**

Naming the ORACLE owner **oracle** is fine for sites with one database administration team. Some sites, especially larger ones, require that the same version of ORACLE be installed two or more times, and that the copies be administered by completely separate DBA teams.

Valid reasons for separating ORACLE software for distinct DBA teams include:

- Enforce security by separating the two software owner logins; one DBA group may not see or alter the other's work.
- The two DBA groups may upgrade, patch, or customize their database software sets at different rates. This can actually be an effective argument for not separating software into two distinct copies.

If your site has multiple DBA teams that own separate copies of ORACLE, distinguish the *oracle* user names according to team or project names. If you foresee two or more DBA teams in the future of your site, you may wish today to choose a different login name than **oracle**. However, since renaming a SunOS login is a simple task, choosing the name **oracle** does not incur a high cost even if some day the name is deemed less than fully appropriate.

We do not recommend that you embed the ORACLE version number in the login name.

The group identification number you select should correspond to a group named **dba** in the **/etc/group** file. The name **dba** is hard-coded into ORACLE's **rdbms/lib/config.c** file. If you change the DBA group name to something other than **dba**, the ORACLE DBA must relink ORACLE during installation.

Although you may specify anything you like in the GCOs field, the BSD **finger(1)** command interprets comma-separated entries in the following order:

- full name
- office number and building
- office telephone extension
- home phone number
- login universe (applicable only on some systems)

Since the *oracle* user is a periodically used responsibility login and not an actual person, it is most reasonable to describe only the login and omit the personal fields.

The *oracle* user's home directory should match the pattern **/*home/oracle**, or you may include an additional group layer in the home directory pattern **/*home/dba/oracle**. This group layer allows you to refer to all members of a given group as a unit.

The *oracle* user has two special privileges that other members of the **dba** group do not have:

- changing the ownership or mode of an ORACLE file
- writing mode 755 database files and directories

Because the properly installed ORACLE system's database files never need ownership or mode adjustment, there is no need to log in as *oracle* except at installation, patch, or upgrade time. We recommend that only the site's master DBA have access to the *oracle* user account.

The editing of mode 755 files like **/etc/oratab** should be coordinated through one user who acts as the site's master DBA. Configuration tuning involving movement of a database file from one disk drive to another should likewise be coordinated through the master DBA, as should creation of files and directories in mode 755 directories like the *oracle* user home directory.

All normal database maintenance, including starting and stopping ORACLE and creating database logbook files can be performed by any member of the **dba** group.

Create personal DBA logins

Each database administrator on your system, including your master DBA, should have their own personal SunOS login.

For each database administrator, create a SunOS login with the following properties:

- login name connotes the person's actual name
- UID number between 3 and 32767
- default GID number between 2 and 32767, corresponding to the group named **dba**
- GCOS field containing full name, office, and telephone information of the user (append **universe(att)** for systems with **universe(1)**)
- home directory matching the pattern **/*/home/login** or **/*/home/dba/login**
- login shell **/bin/ksh** or **/bin/csh**

A DBA's personal login password entry should contain personal information about that administrator.

Create Applications owner login

This section applies to sites which are installing Oracle Applications such as Oracle Financials and Oracle Manufacturing.

The strategy used to separate the ORACLE database file owner into the *oracle* user is useful in applications administration as well.

For each set of applications software, create a unique SunOS login with the following properties:

- login name connotes the application responsibility, e.g., **applmgr**
- UID number between 3 and 32767
- default GID number between 2 and 32767, corresponding to a connotatively named group, e.g., **oaa** (Oracle Applications Administrator)
- GCOS field describing the login, e.g., Oracle Applications owner (append **,universe(att)** for systems with **universe(1)**)
- home directory matching the pattern **/*/home/login** or **/*/home/group/login**
- login shell **/bin/ksh** or **/bin/csh**

Create ORACLE user logins

Create a SunOS login for each ORACLE user with the following properties:

- login name connotes the person's actual user name
- UID number between 3 and 32767
- default GID number between 2 and 32767, corresponding to a connotatively named group, e.g., **clerk**

- GCOS field containing full name, office, and telephone information (append, `universe(att)` for systems with `universe(1)`)
- home directory matching the pattern `/*/home/login` or `/*/home/group/login`
- login shell `/bin/ksh`, `/bin/csh`, or `/bin/rksh`

The SunOS system administrator should never allow multiple users to share a login.

ORACLE Files and Data

A handful of ORACLE files belong in SunOS software directory subtrees. These files control the switching among ORACLE databases or versions of ORACLE software. Since the programs themselves are run to determine user `$PATH` settings, they need to be stored in directories in which all system user programs are prepared to look.

Create ORACLE files

Create the following empty files with the `touch(1)` command:

- `/usr/bin/oraenv`, owner *oracle*, group *dba*, mode *644*
- `/usr/local/bin/coraenv`, owner *oracle*, group *dba*, mode *644*
- `/usr/local/bin/dbhome`, owner *oracle*, group *dba*, mode *755*
- `/etc/oratab`, owner *oracle*, group *dba*, mode *644*
- `/etc/sqlnet`, owner *oracle*, group *dba*, mode *644*
- `/etc/tnsnames.ora`, owner *oracle*, group *dba*, mode *644*
- `/etc/listener.ora`, owner *oracle*, group *dba*, mode *644*

These are the only ORACLE files that need to be in SunOS software subtrees. The DBA will fill in the contents of these files during installation.

The ORACLE files in `/usr/local/bin` and `/etc` are owned by the *oracle* user so that the database administrator can update these files at installation or upgrade time without submitting a request to the SunOS administrator. The blank file scheme makes it possible to do this without giving write permissions on your SunOS software directories to the ORACLE DBA.

The administrative effort and risk of moving, saving, and replacing ORACLE system files in `/usr` is a business expense that can be minimized by limiting the number of ORACLE system files in `/usr`.

The `etc` directory contains system configuration files and some utilities critical to maintaining a SunOS system. The `etc` directory lives in the root partition because its files always need to be available, and the root file system is always mounted. Although a SunOS upgrade may replace some of the utilities in the `etc` directory, configuration files are not disturbed.

Create ORACLE data directories

The following suggestion offers a solution to the problem of how to store massive amounts of ORACLE data on a SunOS system without sacrificing I/O performance.

Make a subdirectory named `/*/ORACLE` (uppercase) in each root-level user-data directory. Give these directories owner *oracle*, group owner *dba*, and mode 770.

This step guarantees complete load balancing flexibility while supplying an organized scheme for separating ORACLE data from all other data and software. By following this suggestion, the DBA gains complete freedom to increase the size of ORACLE tablespaces and optimize disk load balance as experience with the system accumulates.

Load-balancing flexibility by using pattern-named directories across devices applies not only to ORACLE software, but to any product that benefits from distribution of files across multiple disk devices.

An uppercase name exploits the case-sensitivity of SunOS to further distinguish this data directory from the *oracle* user's login home directory.

Having all database data directories occur at the same level in the SunOS directory tree, allows for manipulation of collections of ORACLE files as a unit in any UNIX command.

Using mode 770 ensures that only the *oracle* user and members of the *dba* group can copy and delete database files. Other users are not permitted even to navigate through an ORACLE data directory.

Consider Raw Devices

ORACLE for SunOS provides the option to read and write directly to unmounted SunOS disk sections called raw devices. Decide whether the use of raw devices would be advantageous on your system.

Using raw devices can greatly complicate the administration of both the database and SunOS. Making all sections the same size eliminates some of these complications. The reason all sections need to be the same size is that this is the only way it is possible to move ORACLE database files from one disk to another.

Choosing the right size for all of a system's raw sections can be a challenge. You must make the size large enough that you can fit your data in no more than the maximum number of data files your ORACLE instance can open. On the other hand, you must make the size small enough so you don't waste an appreciable amount of space in tablespaces that don't need to be very large. Picking the right section size is something that both the experienced SunOS administrator and the

experienced ORACLE administrator need to discuss over a picture of the disk hardware partition maps before formatting the disks.

On SunOS, logical volumes or virtual disks are provided, and can alleviate some of the issues associated with raw devices.

For more information on raw devices and logical volumes, see the "ORACLE System Administration" chapter.

ORACLE DBA Recommendations

This section offers suggestions to the ORACLE database administrator with regard to setting up various directory structures for ORACLE software, users, and other DBAs. The steps suggested in this section should be performed after the SunOS system administrator has completed the steps in the previous section.

The following is a brief list of the recommended steps detailed in this section:

Steps for the base directory:

1. Create a base directory for the *oracle* user and set the ORACLE_BASE environment variable.
2. Create \$ORACLE_BASE subdirectories.
3. Create version subdirectories.
4. Set ORACLE_HOME.

Steps for ORACLE files in the SunOS file system:

5. Fill in ORACLE files in SunOS directories.

Steps for user logins:

6. Update user login files.
7. Update DBA login files.

Base Directory

The following figure is an example of the recommended Oracle Base directory structure.

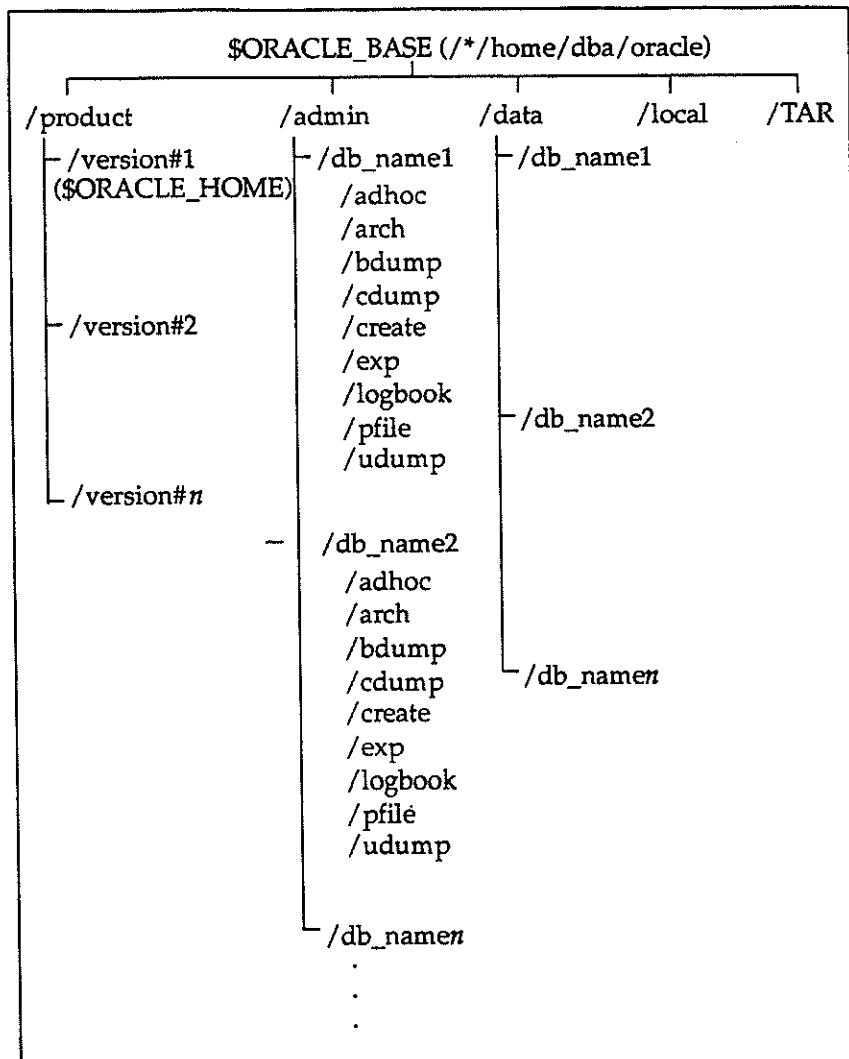


FIGURE 2-2 Recommended \$ORACLE_BASE file tree

Create a base directory

Create a base directory for the *oracle* user to match the pattern **/home/oracle* or **/home/dba/oracle*. Give the *oracle* user base directory mode 755. This directory will be referred to as **\$ORACLE_BASE**. There should be only one **ORACLE_BASE** directory per system.

Mode 755 helps to enforce this recommendation by making it impossible for anyone not logged in as *oracle* to create files in the *oracle* user's base directory.

Create
ORACLE_BASE
subdirectories

Create the following subdirectories in **ORACLE_BASE**:

product	will contain the version subdirectories where you will install ORACLE (\$ORACLE_HOME)
admin	will contain a sub-directory for each named database and house such administrative information as your init.ora file
data	will contain a sub-directory for each named database and house data related information for those databases. In a well-configured production system, this directory should be empty.
local	will contain database scripts, and other miscellaneous files
TAR	will contain information pertaining to technical assistance requests logged with Oracle Worldwide Support

We recommend that the database administration team keep an online record of all Technical Assistance Requests (TARs) logged with Oracle Worldwide Support.

Since TARs are relevant system wide, the TAR directory resides above the version subdirectories.

Give the TAR log directory mode 775, which allows all DBAs to create, modify or delete files there. Other users are permitted to read these files.

Create a version
subdirectory

When you first install ORACLE, you will create only one version subdirectory that will contain the current release of the Oracle distribution. As you progress through upgrade versions of ORACLE, you will create additional version subdirectories. These version subtrees allow you to easily run two or more versions of ORACLE software simultaneously, enabling you to test new software before deciding to upgrade or migrate your production system.

Create a version directory in **\$ORACLE_BASE/product** that matches the pattern ***.*.***, using the first three components of the ORACLE release number. Give the directory owner *oracle*, group *dba*, and mode 755.

Mode 755 helps to ensure that the version subdirectory contains nothing but ORACLE software owned by the *oracle* user.

Set **ORACLE_HOME**

The value of **ORACLE_HOME** in a user's environment determines which set of ORACLE software that user will access.

Set the **ORACLE_HOME** environment variable to a version directory matching the pattern **\$ORACLE_BASE/product/*.***. This is where you will install the current release of ORACLE.

The following output shows the contents of a typical **\$ORACLE_HOME** directory.

```
$ cd $ORACLE_HOME
$ pwd
/u01/home/dba/oracle/product/7.0.12
$ ls -lg
total 34
drwxr-xr-x  6 oracle  dba      2048 Feb 17 14:17 aps
drwxr-xr-x  6 oracle  dba      2048 Nov  8 14:52 async
drwxr-xr-x  2 oracle  dba     4096 Feb 18 16:35 bin
drwxr-xr-x  6 oracle  dba      2048 Nov  8 14:55 crt
drwxr-xr-x  2 oracle  dba      2048 Feb 13 12:23 dbs
drwxr-xr-x 10 oracle  dba      2048 Nov  8 14:56 forms30
drwxr-xr-x  2 oracle  dba      2048 Nov  8 14:52 lib
drwxr-xr-x  8 oracle  dba      2048 Nov  8 14:57 loader2
drwxr-xr-x  4 oracle  dba      2048 Nov  8 14:59 menu5
drwxr-xr-x  3 oracle  dba      2048 Feb 10 21:36 orainst
drwxr-xr-x  6 oracle  dba      2048 Nov  8 14:58 plsqli
drwxr-xr-x 12 oracle  dba      2048 Feb  6 18:41 rdbms
drwxr-xr-x  7 oracle  dba      2048 Nov  8 14:58 report
drwxr-xr-x  7 oracle  dba      2048 Nov  8 14:59 sqlplus
drwxr-xr-x  7 oracle  dba      2048 Feb 11 15:23 sqlreport
drwxr-xr-x  8 oracle  dba      2048 Nov  8 14:59 tcp
```

This recommendation makes minor version number upgrades as simple as switching to new software executables. Major version migrations, such as from Version 6 to ORACLE7 Server are more involved, but this scheme simplifies the process as much as possible.

For example, you can run ORACLE7 Release 7.0.11 in uninterrupted production while you install and test operation with Release 7.0.12. At Release 7.0.12 cutover time, shut down the 7.0.11 instance using the **SQL*DBA shutdown** command. Make sure that all your users use an **\$ORACLE_SID** and **\$ORACLE_HOME** that will connect them with Release 7.0.12 software, and when you start the 7.0.12 instance, cutover is complete. Migrating from Version 6 to ORACLE7 Server requires a similar software switch.

ORACLE Files

Fill in ORACLE files

If the SunOS system administrator has followed the steps suggested in the "SunOS System Administrator" section, it should be possible for the DBA to edit the ORACLE files that reside in the SunOS software subtrees.

Fill in the following files created earlier by the SunOS system administrator:

- `/usr/local/bin/oraenv`: copy `$ORACLE_HOME/bin/oraenv`
- `/usr/local/bin/coraenv`: copy `$ORACLE_HOME/bin/coraenv`
- `/usr/local/bin/dbhome`: copy `$ORACLE_HOME/bin/dbhome`
- `/etc/oratab`: copy `$ORACLE_HOME/bin/oratab` and append an entry for each instance on the system
- `/etc/sqlnet`: create a two-column *alias* and *hostring* entry for each alias on the system
- `/etc/tnsnames.ora`: copy `$ORACLE_HOME/network/config/client/tnsnames.ora` and edit it to contain *alias=hostring* entries for the system
- `/etc/listener.ora`: copy `$ORACLE_HOME/network/config/server/listener.ora` and edit it to contain listener entries for the system



See the SQL*Net V1 Overview chapter in the TARG for information on `/etc/oratab` and `/etc/sqlnet`. See the SQL*Net V2 chapters in the TARG for information on `/etc/tnsnames.ora` and `/etc/listener.ora`.

Login Files

Update user login files

Set each user's `ORACLE_HOME` by sourcing the `oraenv` program from the user's `.profile` file for Bourne or Korn shell, or by sourcing the `coraenv` program from the user's `.cshrc` file for C shell. The `(c)oraenv` program prompts the user for an instance name and reads `/etc/oratab` to determine the correct setting for the user's `ORACLE_HOME` and `PATH` variables.

For example, the master DBA should put the following lines in the `.profile` file of each user in order to access the `sid1` instance as the default:

```
PATH=/bin:/usr/bin:/usr/local/bin
# set default instance
```

```
ORACLE_SID=sid1
# set ORACLE_SID, ORACLE_HOME, and PATH without asking
ORAENV_ASK=NO ; . oraenv ; ORAENV_ASK=
```

A user accessing another instance by default would change only the **\$ORACLE_SID=** line of their **.profile** file. User wishing to use *sid1* as their default could switch to the *sid2* instance by executing:

```
$ . oraenv
ORACLE_SID = [sid1] ? sid2
```

Update DBA login files

In addition to the login file lines supplied for all ORACLE users, database administrators need a few additional items in their login files. One database administrator responsibility is to make database logbook entries describing the structure and state of each ORACLE database on the system. Another is to record TARs logged with Oracle Worldwide Support. To facilitate the performance of these duties, each DBA should set the **ORACLE_PATH** environment variable in his or her login file.

The **ORACLE_PATH** environment variable lists all possible search paths contained in SQL scripts. Setting this variable enables you to change to a logbook directory and execute scripts from another directory without having to reference a long path name.

Each DBA on the system may source (c)**dbaenv** to set **ORACLE_HOME**, **ORACLE_SID**, **ORACLE_PATH**, and **PATH**. Using this (c)**dbaenv** file, an ORACLE DBA would change database instances as follows:

```
$ . dbaenv
ORACLE_SID = [sid1] ? sid2
```

The (c)**dbaenv** file would set the DBA's environment variables in addition to performing the standard duties of (c)**oraenv**.

Preparing to Install ORACLE7 Server

Prepare Your SunOS System

This section lists the minimum preparations that you must make before installing ORACLE7 Server.

Installation Summary Sheet

The following table lists the key directories, files and accounts that are referred to during the installation. For each object in the table, the Default Value shown in bold is the name of the object provided by default; Value on Your System is the actual value you assign to the object on your own system. You should enter these values in the table as a reminder.

If you have implemented the Oracle Base Structure by following all of the recommendations in the previous two sections, use the first table. If you chose not to implement the Oracle Base Structure, use the second table. The difference between the two is that the Oracle Base Structure table includes the **ORACLE_BASE** environment variable, and different default values for the control and data files.

Object	Default Value	Value on Your System
ORACLE base directory (\$ORACLE_BASE)	none	
ORACLE home directory	/user/oracle	
ORACLE owner account	oracle	
ORACLE acct passwd	none	
DBA group name	dba	
database name	none	
system identifier (\$ORACLE_SID)	none	

FIGURE 2-3 Installation Summary Sheet for the Oracle Base Structure

Object	Default Value	Value on Your System
ORACLE home directory (\$ORACLE_HOME)	/user/oracle	
ORACLE owner account	oracle	
ORACLE acct passwd	none	
DBA group name	dba	
system identifier (\$ORACLE_SID)	none	
database name	none	

FIGURE 2-4 Installation Summary Sheet

Tasks

Perform the following tasks and where applicable, enter the appropriate values in the Installation Summary Sheet above. If you have followed the recommendations in the previous sections for the Oracle Base Structure, you have already performed these tasks. Go directly to the next section.

1. Create a home directory for the ORACLE7 Server.
2. Create the *oracle* owner account and password.
3. Make the *oracle* account's HOME directory *ORACLE_HOME*.
4. Create a special group-id for the database administrator.
5. Make both the *oracle* owner and *root* as members of this special database administration group-id.

The Installer assigns special privileges to this group, including access to SQL*DBA and its special "system privileged" functions. After you run the Installer, *oracle* and *root* become "privileged DBA accounts." In addition to making the *oracle* user a member of the *dba* group, you should make *dba* the default group of the *oracle* user.

Note: Even though both the *oracle* and *root* users should belong to the *dba* group, the *oracle* user should *not* be a member of the *root* group. In fact, the *root* user should be the only member of the *root* group.

6. Decide on a value for the ORACLE7 Server system identifier.

This is an alphanumeric “word” used to identify the instance associated with the database that will be created by the Installer. The system identifier will be referred to in this documentation as the *sid* or *ORACLE_SID*.

7. Determine names and locations for your control files.

If you choose to have the Installer create your database for you, you will be prompted for the names and sizes of your control files. By default, all three control files are located in one place. We strongly recommend that each of these files be placed on a different disk to minimize the risk of data loss.

If you are using the Oracle Base Structure, your default control files are: **\$ORACLE_BASE/data/control01.ctl**, **control02.ctl**, and **control03.ctl**.

If you are not using the Oracle Base Structure, your default control files are: **\$ORACLE_HOME/dbs/ctrl1sid.ctl**, **ctrl2sid.ctl**, and **ctrl3sid.ctl**.

8. Determine names and locations for your redo log files.

If you choose to have the Installer create your database for you, you will be prompted for the names and sizes of your redo log files. By default, all three are located in one place. We strongly recommend that each of these files be placed on a different disk to minimize disk I/O contention.

If you are using the Oracle Base Structure, your default redo log files are: **\$ORACLE_BASE/data/redo01.log**, **redo02.log**, and **redo03.log**.

If you are not using the Oracle Base Structure, your default redo log files are: **\$ORACLE_HOME/dbs/log1sid.dbf**, **log2sid.dbf**, and **log3sid.dbf**.

By default, the redo log files are 500K each. The minimum allowable size for the redo log files is 100K each.

9. Determine names and locations for your data files.

If you choose to have the Installer create your database for you, you will be prompted for the names and sizes of your data files. By default, all of your data files are located in one place. We strongly recommend that each of these files be placed in a directory other than **\$ORACLE_HOME**.

If you are using the Oracle Base Structure, your data files will be created as follows:

Default Data File	Default Size	Minimum Size
\$ORACLE_BASE/data/system01.dbf	8M	5M
\$ORACLE_BASE/data/rbs01.dbf	2M	1400K
\$ORACLE_BASE/data/temp01.dbf	550K	260K
\$ORACLE_BASE/data/tools01.dbf	4M	1M
\$ORACLE_BASE/data/users01.dbf	1M	200K

If you are not using the Oracle Base Structure, your data files will be created as follows:

Default Data File	Default Size	Minimum Size
\$ORACLE_HOME/dbs/sytsid.dbf	8M	5M
\$ORACLE_HOME/dbs/rbssid.dbf	2M	1400K
\$ORACLE_HOME/dbs/tempsid.dbf	550K	260K
\$ORACLE_HOME/dbs/toolssid.dbf	4M	1M
\$ORACLE_HOME/dbs/usrsid.dbf	1M	200K

10. Configure the UNIX kernel parameters described in the “Key to ORACLE” chapter.

Relink the UNIX kernel and reboot your system prior to installing ORACLE.

11. Review installation requirements for specific products.

For each product you will install, turn to its “Preparing to Install...” section and make sure you have met its installation requirements.

12. Check for restrictions.

Make sure you are familiar with all the information listed in the “SunOS-Specific Restrictions” and the “Restrictions and Workarounds” sections in the “Key to ORACLE” chapter.

Preparing to Install Oracle Precompilers

Prior to installing new Oracle Precompilers, you should save your old application source files before deinstalling the old precompiler distribution. Also, make sure no one is using a precompiler before deinstalling. Refer to the section on the **deinstall** utility in the "Upgrading and Migrating your ORACLE System" chapter for instructions.

Pro*Ada

Upgrading Pro*Ada

If you are upgrading Pro*Ada from V1.3, changes in directory structure require you to change any pointers to the previous Pro*Ada directory (**\$ORACLE_HOME/ada**) which may be included in your make files, to **\$ORACLE_HOME/proada**. Also note that the executable for Pro*Ada has been changed from **pcc** to **proada**.

Before installing Pro*Ada, you must add the full name of the bin directory of the Sun Ada compiler to the **path** environment variable. For example, if your Sun Ada compiler resides under **/usr/lib/ada**, enter (for C shell):

```
set path=($path /usr/lib/ada/bin)
```

You must also set the **LM_LICENSE_FILE** environment variable to point to the Sun Ada license file.

Pro*C

Upgrading Pro*C

If you are upgrading Pro*C from V1.3, changes in directory structure require you to change any pointers to the previous Pro*C directory (**\$ORACLE_HOME/c**) which may be included in your make files, to **\$ORACLE_HOME/proc**. Also note that the executable for Pro*C has been changed from **pcc** to **proc**.

Pro*COBOL

Upgrading Pro*COBOL

If you are upgrading Pro*COBOL from V1.3, changes in directory structure require you to change any pointers to the Pro*COBOL V1.3 directory (**\$ORACLE_HOME/cobol**) which may be included in your make files, to **\$ORACLE_HOME/procob**. Also note that the executable for Pro*C has been changed from **pcc** to **procob**.

Pro*FORTRAN

Upgrading Pro*FORTRAN

If you are upgrading Pro*FORTRAN from V1.3, changes in directory structure require you to change any pointers to the Pro*FORTRAN V1.3 directory (**\$ORACLE_HOME/fortran**) which may be included in your make files, to **\$ORACLE_HOME/profor**. Also note that the executable for Pro*FORTRAN has been changed from **pcc** to **profor**.

Preparing to Install SQL*Net Asynchronous

The SQL*Net Asynchronous User Account

Before installing SQL*Net Asynchronous, create a user account (the default name is **sqlnet**), with a password and home directory.

Deinstalling SQL*Net Asynchronous

If you are upgrading SQL*Net Asynchronous, you must deinstall the old version before installing the new one. You can deinstall the product manually, or use the **deinstall** utility.

Refer to the section "Deinstalling Old Distributions" in the "Installation Options" chapter for instructions for either procedure.

Before deinstalling:

- users must back up their customized dialog files (**unix.dia**, **unix.lgo**, **unix.cli**, **unix.lgf**)
- be sure no one is using SQL*Net Asynchronous

Preparing to Install SQL*Net TCP/IP

Deinstalling SQL*TCP/IP

An application uses the TCP/IP network to connect to the **orasrv** service on a remote machine. The **orasrv** service receives the sid from the application and looks up the **ORACLE_HOME** for the database in the **/etc/oratab** file. It then sets up the environment and invokes the **oracle** shadow process.

Configure the Listener

It is recommended that you reserve a port for your SQL*Net V2 listener in the **/etc/services** file of each node on the network that defines the SQL*Net V2 listener port. The port is commonly 1521. The entry should list the listener name and the port number. For example:

```
listener                1521/tcp
```

where "listener" is the name of this particular listener, as set in the **listener.ora** and **sqlnet.ora** files.

Preparing to Install SQL*Net V2

Overview

When you install SQL*Net V2, you must also install at least one protocol adapter at the same time and relink the TNS listener. The Installer will prompt you for each available protocol adapter during the installation. If you choose to install multiple protocol adapters during one Installer session, you need only relink the TNS listener once. If you have previously installed SQL*Net V2 and a protocol adapter (which happens automatically when you install ORACLE7 Server), and now want to install an additional protocol adapter, you need not reinstall SQL*Net V2 or relink the TNS listener. In such a case, you may also be installing a MultiProtocol Interchange.

Read the SQL*Net V2 README.doc File

The README.doc file, located in the \$ORACLE_HOME/network/doc directory, provides a list of changes in SQL*Net V2 since the last released version. This file, which outlines administrative procedures and describes the latest known restrictions, serves as an on-line supplement to the printed documentation.

Install Protocol Software & Hardware

When you install an Interchange, at least two Oracle protocol adapters should be installed. (You can install them prior to installing the Interchange.) Protocol adapters require their corresponding vendor software and hardware. Refer to the requirements charts in the "Key to ORACLE" chapter for requirements. Refer to these vendors' documentation for installation and configuration information.

If not already installed, install the appropriate protocol hardware and software on each node in the network for each protocol community.

Verify that the vendor's software is working properly. A simple file transfer to and from the host should be sufficient.

Preparing to Install an Interchange

Overview

Installing an Interchange requires SQL*Net V2 to be already installed. This condition is satisfied whether you install SQL*Net V2, the protocol adapters, and the Interchange all at one sitting, or piecemeal, as long as you install SQL*Net V2, the protocol adapters, and the Interchange, in that order. The instructions given in this section supplement the instructions for installing ORACLE (in the "Installing Oracle Products" chapter) and SQL*Net V2 (in the "Installing and Configuring SQL*Net V2" chapter).



This section focuses on preinstallation activities. It assumes you have already digested the information in the rest of the SQL*Net V2 documentation set, and have planned your network communities, including which nodes will be designated as Interchanges. Refer especially to Chapter 4 in the *MultiProtocol Administrator's Guide* for information on planning a multi-protocol network and its specific setup requirements and tasks.

Deinstall the Interchange

If you are upgrading from an earlier version of the MultiProtocol Interchange, you must deinstall the old version before installing the new one. You can either deinstall the product manually, or use the **deinstall** utility. Refer to the "Deinstalling Old Distributions" section in the "Upgrading and Migrating your ORACLE System" chapter for either procedure.

Before deinstalling:

- you may want to save the network configuration and trace files you have built (**intchg.ora**, **tnsnet.ora**, and **tnsnv.ora**)
- be sure no one is using the Interchange

Installing only the Navigator component

Normally, you will install both components of an Interchange — the Connection Manager and the Navigator. However, if you have a very complicated network consisting of many different protocol communities, you may designate additional nodes to be "client Navigators," on which you install only the Navigator.

To install the Navigator only, select **Y** to the first prompt to install the Interchange. You are then prompted to install the individual Interchange components. Answer **Y** only to installing the Navigator.

Relinking

The Installer has two prompts for relinking: once for the Interchange and once for the Navigator.

Relink *both* components when:

- installing an Interchange
- installing a new protocol adapter

Relink *only* the Navigator when installing only a Navigator.

Deinstalling the DECnet Adapter

If you are upgrading the DECnet adapter, you must deinstall the old version before installing the new one. You can deinstall the product manually, or by using the **deinstall** utility.

Refer to "Deinstalling Old Distributions" in the "Installation Options" section for instructions.

Before deinstalling, be sure no one is using the DECnet adapter.

Typical Administrative Tasks

To prepare for installation of SQL*Net DECnet with SQL*Net V2, your UNIX-based operating system administrator must perform the following tasks.

1. **Install vendor-specific software.**
2. **Define all accessible network nodes using the network utilities supplied with your software.**
3. **Create a default user account with a password and home directory.**
4. **Verify that your protocol adapter is functioning properly.**

SQL*Net TCP/IP

Configure the Listener

It is recommended that you reserve a port for your SQL*Net V2 listener in the `/etc/services` file of each node on the network that defines the SQL*Net V2 listener port. The port is commonly 1521. The entry should list the listener name and the port number. For example:

```
listener          1521/tcp
```

where "listener" is the name of this particular listener, as set in the `listener.ora` and `sqlnet.ora` files.

Preparing to Install Office Automation Products

Oracle *Mail

If you are upgrading an existing Oracle*Mail system, shut down Oracle*Mail completely before you begin. See the *Oracle*Mail Administrator's Guide* for details.

CHAPTER

3

INSTALLING ORACLE PRODUCTS

This chapter guides you through installation of ORACLE7 Server and related products on SunOS. The information contained in this chapter is for users who are installing ORACLE7 Server for the first time. See the "Where to Begin" section of the "Preparing to Install Oracle Products" chapter for directions if you are upgrading or migrating from a previous release.

This chapter covers the following topics:

- Installer overview
- how to use the Installer
- the Install Actions menu
- mounting a CD-ROM and starting the Installer
- installation messages
- completing the installation

Installer Overview

What is the Installer?

The Installer is an interactive menu-based program that provides the ability to:

- check product dependencies and disk space
- install Oracle products quickly and conveniently
- answer all questions about installing before the actual installation takes place
- update existing Oracle products to the latest version
- remove existing Oracle products
- list available and installed Oracle products
- access a comprehensive online help facility

Task Overview

There are two basic phases to the installation: loading the Oracle distribution, and installing the distribution. The Installer allows you to perform both of these phases in one or more sessions, depending on your needs.

The figure below shows the main tasks you will carry out during installation.

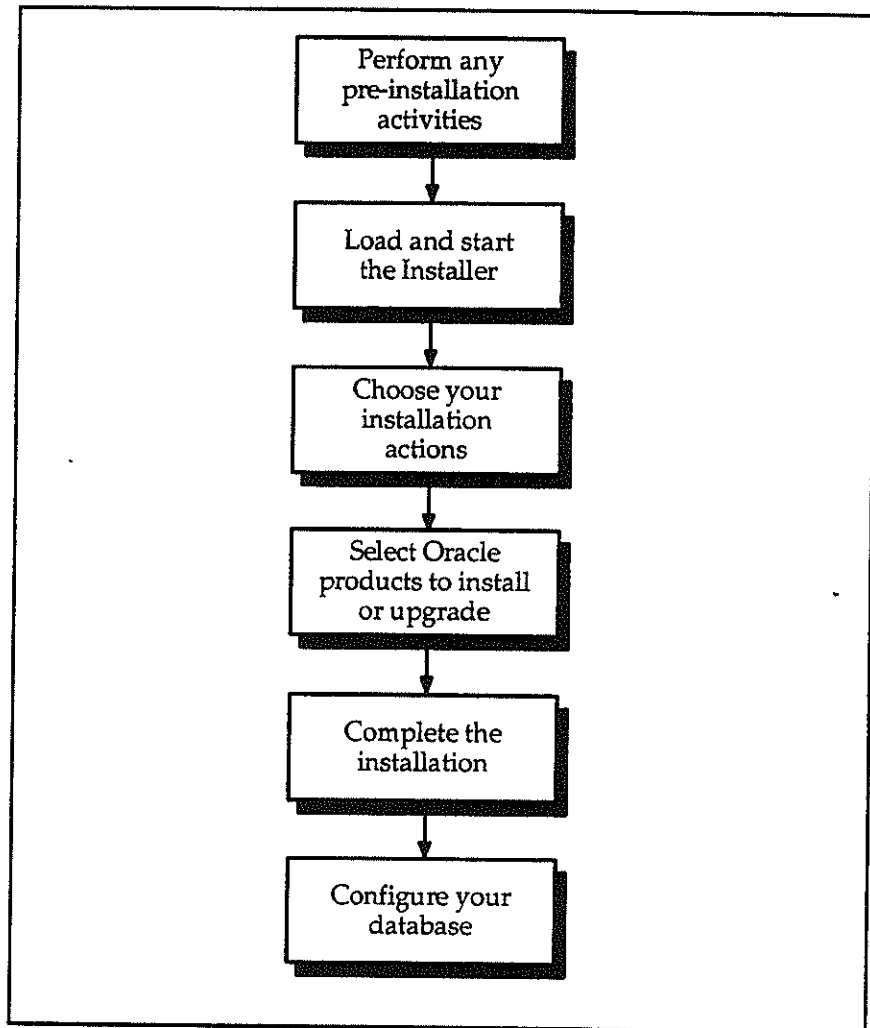


FIGURE 3-1 Installer Task Overview

Online Information	After you read in the Oracle distribution, you can access the following files, which contain supplemental information. Use your SunOS editor to read these files.						
ORACLE7 Server files	The following files are found in the \$ORACLE_HOME/rdbms/doc directory: <table> <tr> <td>readmeunix.doc</td><td>This file contains SunOS-specific information on the current version of ORACLE7 Server, including supported terminals, and a SunOS-specific "fixed bug list" with brief descriptions.</td></tr> <tr> <td>README.doc</td><td> This file serves as an online addendum to the RDBMS documentation set and contains updates such as new initialization parameters, information on SQL*DBA, ORACLE utilities, security, and notes on preparing for future releases. It also includes a "fixed bug list," which briefly describes all the restrictions that have been fixed in this and recent releases. These bugs are identified by numbers referred to by Oracle Worldwide Support. </td></tr> <tr> <td>readme.dat</td><td>This file contains online instructions for using the CD-ROM distribution.</td></tr> </table>	readmeunix.doc	This file contains SunOS-specific information on the current version of ORACLE7 Server, including supported terminals, and a SunOS-specific "fixed bug list" with brief descriptions.	README.doc	This file serves as an online addendum to the RDBMS documentation set and contains updates such as new initialization parameters, information on SQL*DBA, ORACLE utilities, security, and notes on preparing for future releases. It also includes a "fixed bug list," which briefly describes all the restrictions that have been fixed in this and recent releases. These bugs are identified by numbers referred to by Oracle Worldwide Support.	readme.dat	This file contains online instructions for using the CD-ROM distribution.
readmeunix.doc	This file contains SunOS-specific information on the current version of ORACLE7 Server, including supported terminals, and a SunOS-specific "fixed bug list" with brief descriptions.						
README.doc	This file serves as an online addendum to the RDBMS documentation set and contains updates such as new initialization parameters, information on SQL*DBA, ORACLE utilities, security, and notes on preparing for future releases. It also includes a "fixed bug list," which briefly describes all the restrictions that have been fixed in this and recent releases. These bugs are identified by numbers referred to by Oracle Worldwide Support.						
readme.dat	This file contains online instructions for using the CD-ROM distribution.						
Product files	These files are found in the ORACLE_HOME/product/doc directories, where <i>product</i> is the name of the product's directory. <table> <tr> <td>README.doc</td><td>The product README.doc files detail such topics as changes since the last release, bug fixes, and restrictions.</td></tr> </table>	README.doc	The product README.doc files detail such topics as changes since the last release, bug fixes, and restrictions.				
README.doc	The product README.doc files detail such topics as changes since the last release, bug fixes, and restrictions.						

How to Use the Installer

You can now use the Installer anytime to install, update, list, or remove Oracle products. The following topics are described in this section:

- terminology
- navigation
- function keys
- Product Installation screen
- online help

Terminology

The following concepts and terms are used in this chapter:

Button	an option that appears as a labeled rectangle on the screen. Common buttons used in the Installer are (Install), (Help), (Go), and (Cancel). Buttons are indicated by parentheses throughout this chapter.
Command	an option found in the Pull-down Menu that you select to execute an action
Field	a block within a window where you can enter required information
Installer	an ORACLE executable file called orainst that allows you to install, update, or remove any product for SunOS
Product List	an alphabetical list of Oracle products available to you through the Installer
Pull-down Menu	a list of options that runs along the top of a window. Selecting an item in the Pull-down Menu displays a list of commands or sub-menus
Screen	a rectangular workspace where you view the Installer functions
Window	a self-contained block of information within a screen. Each window in the Installer has a title bar that is highlighted when active

Navigation Keys

Use the following keys to navigate through the Installer screens:

<i>Key</i>	<i>Action</i>
[TAB]	moves you forward from field to field within the screen
[RETURN]	serves as an "Enter" function to invoke an action and proceed to the next screen
[SPACE BAR]	selects or deselects products from the Available and Installed Products lists
[Arrow Keys]	move you up and down through a list or menu

Commands

The following is a list of commands you should be familiar with during the installation:

<i>Command</i>	<i>Action</i>
Shell	suspends the Installer session so that it can be resumed exactly as you left it. This command is available only in the Product Installation screen and can be invoked from the File Menu or by entering [CTRL]l. To resume the Installer, type exit at your system prompt.
Quit	exits the Installer. This command is only available from the Product Installation screen by pressing [T] in the File Menu or by entering [CTRL]t.
(Cancel)	exits the Installer session. Invoke (Cancel) by selecting the button, when available, and pressing [RETURN]. If the (Cancel) button appears in a warning message screen, it serves only to cancel the message, not the Installer session.
Interrupt	exits the Installer session and can be executed at any time by pressing [CTRL]c
Restart	begins a new Installer session without having to exit to the operating system and reinvoke orainst . The Restart command is available only in the Product Installation screen from the File Menu.

Function Keys

A small set of function keys is available so that you can quickly perform common tasks in the Product Installation screen. These function keys are determined by the **TK2DEV** environment variable and are displayed in the Status window in the Product Installation screen.

<i>Key Name</i>	<i>Function</i>
Cancel	cancels an action invoked by another function key
Menu	activates the Pull-down Menu in the upper left corner of the screen
MoveWindow	enables you to reposition the three windows of the Product Installation Screen. This function is described in detail below.
NextWindow	activates the next window in the Product Installation screen. The three windows are cycled through in the following order: Available Products, Installed Products, Status.

If you have set your **TK2DEV** variable to **vt100**, the function keys are listed on the Installer screens. See the table below for a mapping of VT100 function keys to the other available resource files.

Function	vt100	sun	sunx	ansi	at386	ncd220	hft/hftc	hpterm	wyse
Cancel	PF4	R4	R4	F4	F4	PF4	F4	F8	F4
Menu	KP0	[Esc] 1	[Esc] 1	F5	F5	PF0	F8	F4	F5
Move-Window	PF3	R3	R3	F3	F3	KP3	F3	F3	F3
Next Window	PF2	R5	R2	F2	F2	PF2	F2	F2	F2

FIGURE 3-2 Function Key Mappings

Online Help

The Installer's Online Help system is your most valuable tool when using this application. Online Help can be accessed in two ways:

- from each input screen
- from the Pull-down Menu

Accessing Help from the input screen

Each input screen has a (Help) button which you can activate. Use [TAB] to navigate to the (Help) button, and press [RETURN] to activate the Help facility. Accessing Help from the input screen provides you with a context sensitive Help window. The following screen shots are an example of invoking Help from an input screen:

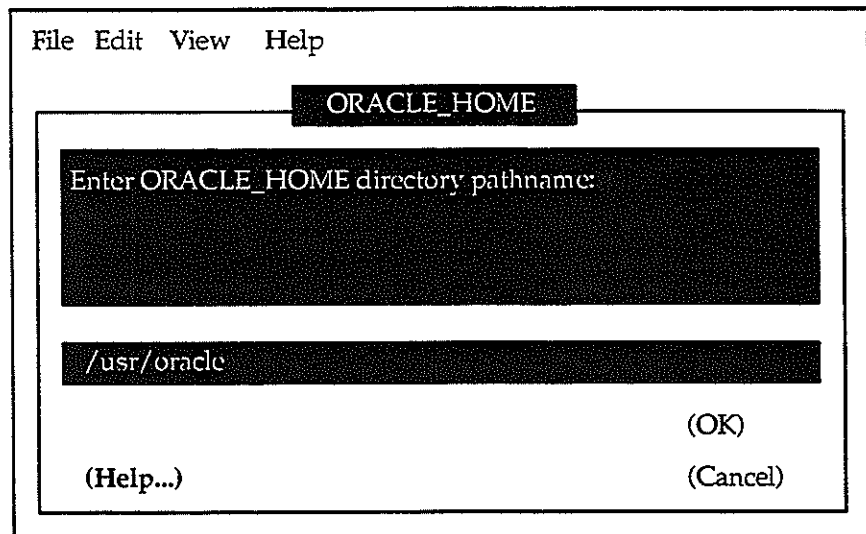


FIGURE 3-3 Installer Sample Input Screen

In the window above, the user selects the (Help) button. The Help screen on the next page is then displayed.

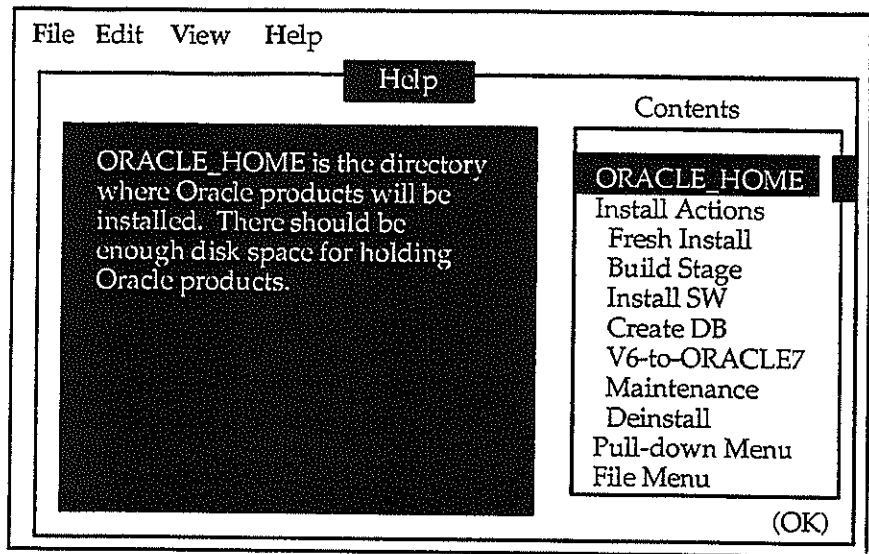


FIGURE 3-4 Installer Online Help

The help screen above not only gives you information on the screen from which you invoked it, but also gives you a menu of additional Installer topics which you can scroll through using the arrow keys. While the first item on the Contents menu will always be context sensitive, the remaining menu options remain the same.

Accessing Help from
the Pull-down Menu

Invoking Help from the Pull-down Menu gives you a submenu of three topics.

Using Installer	this option gives you the same menu of topics as you would get from invoking Help from a particular screen
Product Information	this option gives you a menu of Oracle products to scroll through, providing a brief description of each one
Installer Information	this option gives you the version number of the Installer

Install Actions Screen

After you provide the information in the initial Installer input screens, the Install Actions screen is displayed.

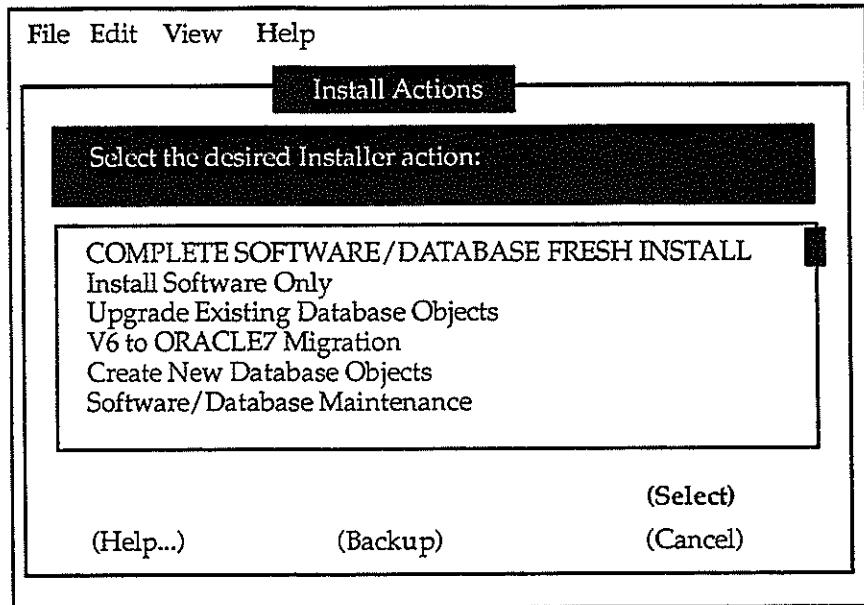


FIGURE 3-5 Install Actions screen

You can select one of the actions using the vertical arrow keys to scroll to and [SPACE BAR] to select or deselect an option. Then use [TAB] to navigate to the (Select) button and press [RETURN] to activate the selection.

The options in the Install Actions screen are discussed individually and in detail in the section titled "The Install Actions Menu."

Product Installation Screen

The Product Installation screen consists of three windows and a Pull-down Menu as shown in the figure below.

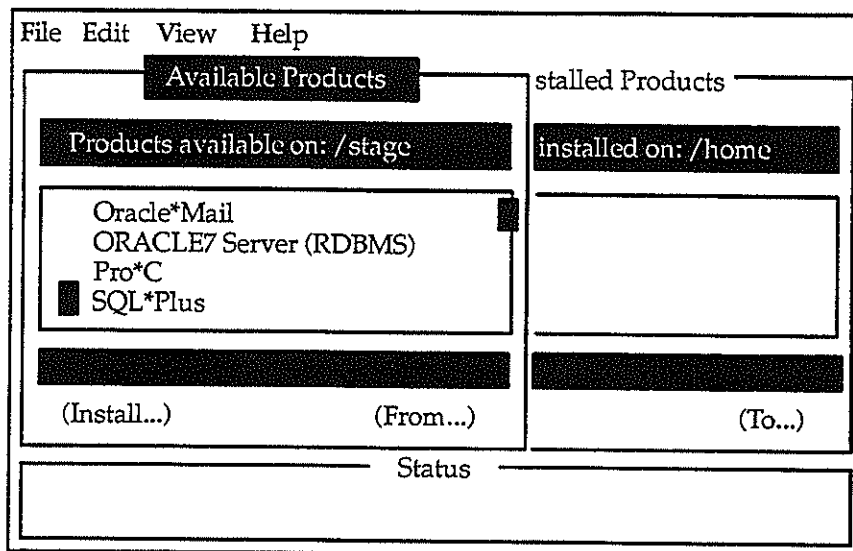


FIGURE 3-6 Installer Product Installation Screen

Available Products window

The Available Products window lists all the Oracle products available for installation. When the Available Products window is active, use the arrow keys to scroll to, and [SPACE BAR] to select, one or more Oracle products. Then, use [TAB] to navigate to the (Install) button and press [RETURN] to activate the installation phase. To "Select All" enter [CTRL-A].

Installed Products window

The Installed Products window, which appears partially hidden above, displays the name of all the Oracle products currently installed. When the Installed Products window is active use the arrow keys to scroll to and [SPACE BAR] to select one or more Oracle products. Then use [TAB] to navigate to the (Remove) button and press [RETURN] to activate deinstallation. To "Select All" enter [CTRL-A].

Status window

The Status window provides on-screen information about the Installer's activities depending on what is currently highlighted in the active window. Function keys for your terminal type are displayed in the Status window.

Pull-down Menu

The Pull-down Menu is only accessible from the Product Installation screen and consists of four menus to help you use the Installer:

<i>Menu Name</i>	<i>Commands</i>
File	contains the Restart, Shell, Quit, and Install From commands
Edit	contains the Select All and Deselect All commands
View	contains the Arrange command as well as a toggle switch for the display of the Available Products window, the Installed Product window, and the Status window
Help	contains the Using the Installer, Product Information, and About the Installer commands

Arranging the windows

By default, the Product Installation screen is designed to appear in a standard window or terminal screen and will be displayed as the above figure. If your window size is larger, you can adjust the display of the three windows in the Product Installation screen with the [MoveWindow] function key.

From the active window, press the [MoveWindow] function key which is displayed in the Status window. A cursor appears in the upper left corner of the active window. Use the arrow keys to move that cursor to the desired new location of the window and press [RETURN]. The upper left corner anchors itself to the cursor.

To restore the original position of the three screens, use the Arrange command from the View Menu in the Pull-down Menu.

The Install Actions Menu

Given the extensive scope of operations you can use the Installer to perform, we recommend that you focus on the exact tasks you want to use the Installer to accomplish. Below is a brief description of the options on the Install Actions menu.

Overview of Install Actions

Complete SW/DB Fresh Install

This option is a quick way to install Oracle software and create a new running database instance in one Installer session. Database configuration flexibility is somewhat limited in that you cannot change the number of control files, or data files. The Installer does not provide you with the ability to create additional tablespaces or customize your `init.ora` file. That must be done manually.

Install Software Only

This option installs ORACLE7 Server and Oracle products without creating a database. Choose this option if you want to create your own customized database with a `create database SQL` script.

Create New DB Objects

This option creates a new database and new product database objects such as help and demonstration tables for the selected Oracle products and their dependent products. Use this option in conjunction with the Install Software Only option to perform product database-related activities.

Upgrade Existing DB Objects

Use this option in conjunction with the Install Software Only option to upgrade ORACLE7 environments that have been installed with the Installer. When this option is used, the Installer prompts for each selected product if you want to create new or upgrade existing database objects.

The table that follows provides guidelines for which action you may wish to take.

If You Want To....	Choose This Option From the Installer Actions Menu
Install Oracle distribution and create a database in a new \$ORACLE_HOME	COMPLETE SOFTWARE/DATABASE FRESH INSTALLATION
Install Oracle software without creating a database	Software Only Installation
Install an individual Oracle product	Software Only Installation
Create a new database	Create New Database Objects
Create new product tables with product dependencies turned on (prompted for dependent products, if any)	Create New Database Objects
Create new product tables with product dependencies turned off	Software/Database Maintenance: Create Product Database Objects
Upgrade from a previous release installed with the Installer (i.e. 7.0.11)	Install Software Only then Upgrade Existing DB Objects
Migrate from V6 using the migration utility	V6 to ORACLE7 Migrate

FIGURE 3-7 Decision chart for Installer Actions

Mounting a CD-ROM and Starting the Installer

Note: The Oracle7 CD-ROM is in Rock Ridge Format, which is supported by SunOS 4.1.2 and 4.1.3 only.

To mount your CD-ROM drive, perform the following steps:

1. **Log in as the root user and create an empty directory `"/cdrom"` for mounting the CD drive.** This is referred to as the "mount point" directory. To do this, enter:

```
#mkdir /cdrom
```

2. **Mount the CD-ROM drive to the mount point directory with the following command:**

```
#mount -r -t hfs /dev/sr0 /cdrom
```

Logging Your Installation

The installation log file provides a record of the installation process, including any errors that may occur during your Installer session. If you should need to contact your support representative, the installation log file will provide an accurate and comprehensive record of your installation.

The Installer will prompt you for a log file name. The default log file name is `$ORACLE_HOME/orainst/install.log`. You can either accept the default or provide your own name, depending on your needs. If you choose to rename the log file, you must enter the full directory path name at the prompt.

You can view the log file as it writes, in another window or on a separate terminal, depending on your environment. Enter the following command at the system prompt to view the installation log file:

```
$ tail -f $ORACLE_HOME/orainst/install.log
```

If you have renamed the installation log file, replace the string above with the full directory path name you have assigned.

It is recommended that you maintain a backup of each installation log file. During later Installer sessions, you are given the option of either appending to or renaming an existing log file.

Starting the Installer

After you have mounted the CD-ROM, use the following steps to start the Installer. You need only run the Installer as the *oracle* user. All *root* actions are performed after the Installer has completed the installation. See the "Completing Your Installation" section for details.

1. Set environment variables.

Before you start the Installer you must set the following environment variables:

ORACLE_HOME	defines the home directory for the <i>oracle</i> account
ORACLE_SID	defines the system identifier (the <i>sid</i>)
TK2DEV	specifies the terminal definition resource file to be used with the Installer and SQL*DBA. If you do not specify a value for this variable, it defaults to VT100.

If you are running *xterm*, you need to bring up *xterm* with the "-sf" option and set **TK2DEV** to *xsun*. If you are running *cmdtool* or *shelltool*, set **TK2DEV** to *sun*.

If you have decided to use the Oracle Base Structure, described in the "Preparing to Install Oracle Products" chapter, you must also set **ORACLE_BASE** in your environment.

For a table of available **TK2DEV** values, see the "Function Keys" section above.

Below is an example of how your environment variables might be set for C shell (*csh*) users. You should place these environment settings in your *.login* or *.profile* file and **source** the file to set the variables; or you can set the variables at the system prompt:

```
setenv ORACLE_HOME /usr/oracle
setenv ORACLE_SID sid
setenv TK2DEV xsun
```

2. Start the Installer.

For CD-ROM distributions, enter the following command from the CD-ROM *orainst* directory, **/cdrom/oracle/orainst**:

```
% ./orainst
```

This performs the necessary setup operations for the Installer, then invokes the Installer for you.

Troubleshooting the Installer

If you encounter any trouble with the Installer, you can access an online file of troubleshooting tips by entering the following command.

```
$ orainst -h
```

Oracle Core Products Installation Messages

This section lists by product, prompts you may receive during your installation. You will not necessarily see all the prompts listed here, and may encounter some that are not included, depending on which products you are installing. Online help is accessible for all these prompts, and may assist you with your decisions.

SQL*Plus

Would you like to load the SQL*Plus Help Facility?

Would you like to load the SQL*Plus Demo Tables?

SQL*Forms

Would you like to install SQL*Forms 3.0 demo forms and tables?

Would you like to relink SQL*Forms 3.0 with PL/SQL?

Would you like to relink SQL*Plus with SQL*Forms 3.0?

SQL*Menu

Would you like to install SQL*Menu 5.0 demo forms and tables?

Would you like to relink SQL*Menu 5.0 with PL/SQL?

Would you like to relink SQL*Forms 3.0 with SQL*Menu 5.0?

SQL*ReportWriter

Please select one of the following as a default terminal type for SQL*ReportWriter:

- srw_vt100 - VT100 all keys available
- srw_100f - VT100, similar to SQL*Forms
- srw_220 - VT220, with special graphics font
- srw_220e - VT220 emulators w/o special font
- srw_320 - VT320 with special graphics font
- srw_340 - VT340 with special graphics font
- srw_ansi - SCO ansi console
- srw_at386 - AT&T or ISC AT386 console
- srw_att610 - AT&T 610, 615, and 620 terminals
- srw_axv300 - DG AviiON workstation
- srw_d216 - VT100 emulation mode
- srw_d412 - VT220 emulation mode
- srw_d462 - VT220 emulation mode

```

srw_d200n - D200 terminal [DG mode]
srw_d216n - D216 terminal [DG mode]
srw_d410n - D410 terminal [DG mode]
srw_hp - HP terminal or hpterm
srw_hpt - HP terminal w/o line-drawing
srw_hpx - xterm
srw_iris-ansi - Iris-Ansi most keys available
srw_n101 - NCD, 101 KB
srw_next - NextTerminal, VT100
srw_cmcae - Communicae, vt100 emulator
srw_sun - Sun shelltool
srw_tm220 - Motorola TM220
srw_tm228 - Motorola TM228
srw_tm228i - Motorola TM228I
srw_tm229 - Motorola TM229
srw_tm229i - Motorola TM229I
srw_vtapo - Apollo vt100 emulator
srw_wyse50 - WYSE 50 terminal
ansiu - ANSI terminal
ansixu - ANSI xterm [SCO Open Desktop]
at386x - UnixWare terminal
at386xu - AT386 xterm
att600u - AT&T 605, 610, 615, 620, 705 terminals
att6103u - AT&T 610, 615, 620 with 103-key keyboard
att630u - AT&T 630 terminal
att730u - AT&T 730 terminal
att4425u - AT&T 4425 terminal
att6386u - AT&T 6386 console
v103_au - V103 Ascii, similar to SQL*Forms 3.0
v102u - V102, similar to SQL*Forms 3.0
vt100u - DEC VT100 terminal
vt220u - DEC VT220 terminal
wy60u - WYSE 60 terminal

```

Would you like to link the SQL*ReportWriter demo user exits?

Would you like to load SQL*ReportWriter tables?

Would you like to centralize SQL*ReportWriter tables?

Would you like to load SQL*ReportWriter demo tables?

Please enter the default maximum pages allowed for SQL*ReportWriter
[0 = no limit]:

Please enter the username for the SQL*ReportWriter demo account:

Please enter the password for the SQL*ReportWriter demo database
account:

Would you like to load the TERMDEF utility's data into the SYSTEM
account for SQL*ReportWriter?

SQL*Net V2 Installation Messages

SQL*Net V2

Would you like to install the SQL*Net Configuration Tool tables in
your database?
Would you like to install *protocol_adapter*?

Oracle Office Automation Installation Messages

Oracle*Mail

Are you licensed to run the Oracle*Mail/UNIX Gateway?

Please select one of the following as a default terminal type for Oracle*Mail:

- DEC VT100
- DEC VTT220
- Sun Workstation
- IBM Color PC
- DEC VT320
- AT&T 605, 610, 615, 620 or 705
- AT386 Console
- AT386 xterm
- UnixWare Terminal
- WYSE 60
- hpterm without line-drawing
- HP terminal in hpmode or hpterm
- HP xterm terminal emulator
- Apollo VT100 emulator
- Motorola TM228
- Motorola TM228i
- DG AVX300, VT100 emulation mode
- NCD 101
- Motorola TM229
- Motorola TM229i
- d216, VT100 emulation mode
- d412, VT220 emulation mode
- d462, V220 emulation mode
- d200, DASHER D200 in native mode
- d216, DASHER D216 in native mode
- d410, DASHER D410 in native mode
- SGI IRIS ANSI
- ANSI Universal
- Motorola TM220

Completing the Installation

Perform the following steps to customize your ORACLE system:

- run the **root.sh** script
- modify **init.ora** parameters
- enable archiving
- edit the user startup files
- ensure orderly startup and shutdown of ORACLE7 Server upon operating system startup and shutdown

Once you have completed these steps, proceed to the “Configuring your ORACLE System” chapter.

Also, check the “Product Updates” chapters for any product specific post-installation activities required for the products you have installed.

Run the root.sh script

The Installer creates a **root.sh** file in **\$ORACLE_HOME/orainst** directory. It must be run by the **root** user when you have completed a successful installation. This shell script sets the necessary permissions for certain Oracle products, and can be reviewed prior to running. To run **root.sh**, enter the following:

```
# ./root.sh
```

Modify init.ora Parameters

The Installer creates a default **init.ora** configuration file that is read each time you start an instance of a database. The parameters listed in this file are used to start the database system. These parameters create a system global area (SGA) that is tuned for a default environment. The parameter values set by this file will often be adequate for your system, but you can adjust these values if you wish.



If you wish to fine-tune your system for optimal performance, refer to the *ORACLE7 Server for UNIX Administrator's Reference Guide (SARG)*. For tuning information specific to SunOS, see the “ORACLE System Administration” chapter in this manual.



You may wish to adjust the SGA parameters to reflect memory limitations and the maximum number of users to access your ORACLE system at one time. See the *SARG* for a description of the SGA, and for the default **init.ora** file that is distributed with this release. See the *ORACLE7 Server Administrator's Guide* for a general description of individual SGA parameters and the **init.ora** file.

In order to check the size of your current SGA, enter the SQL*DBA command **show sga**.

Enable Archiving

Archiving is the key to protecting your database against media failure and other problems. It does not slow down processing, and requires only the disk space necessary to store your archives. You can use the following script in SQL*DBA to turn on archiving:

```
# Enable archive logging.
connect internal
alter database close;
alter database dismount;
alter database mount exclusive;
alter database archivelog;
archive log list;
disconnect
shutdown
startup
```



For information on automatic log archiving to tape, refer to the *SARG*. For more general log file information, refer to the *ORACLE7 Server Concepts Guide*.

Edit the User Startup Files

A prototype database assignment procedure for Bourne shell users is created during the installation as the **.profile** file. A similar prototype for C shell users is created during the installation as the **.login** file (or in some cases, the **.cshrc** file).



Each user account used to access your ORACLE system must have its environment conditioned properly to establish the ORACLE system environment variables at login time. To do this, make sure that its startup file calls **oraenv** (or **coraenv**). In the "ORACLE System Architecture" chapter in the *SARG*, the "**oraenv**" section explains in more depth the role of these important shell scripts.

To call **oraenv** (or **coraenv**), add a line like the following as the last line of each user's **.profile** (or **.login** or **.cshrc**) file. (Substitute the appropriate path for **/usr/local/bin**.)

- **Bourne shell users:** Edit the **.profile** files of all ORACLE users who use the Bourne shell, to contain this command:

```
. /usr/local/bin/oraenv
```

- **C Shell users:** Edit the `.login` (or `.cshrc`) files of all ORACLE users who use the C shell, to contain this command:

```
source /usr/local/bin/oracleenv
```

This will allow all of your users to access your ORACLE system and gain some necessary system-wide information, without requiring several specific modifications to their `.profile` files. On login, users will be prompted for the *sid* of the database desired.

Automatic Database Startup and Shutdown

Two shell scripts are available to ensure clean, automatic database shutdown and startup when your SunOS system is shut down and restarted. These are `dbstart` and `dbshut`, and they exist in the `$ORACLE_HOME/bin` directory after you have read in the ORACLE software. The `dbstart` script ensures a clean startup of all specified ORACLE databases even after a system failure. The `dbshut` script ensures a clean shutdown of all specified ORACLE databases whenever the operating system is shut down.

General Notes:

- The **dbstart** and **dbshut** scripts will bring up or close down all of the databases listed in your **/etc/oratab** file that have the third field specified as Y (yes).

Since both **dbstart** and **dbshut** use the same field in the **/etc/oratab** file, they *must* act upon the same databases. For example, you cannot specify that **dbstart** will automatically start up databases *sid1*, *sid2*, and *sid3*, but **dbshut** will shut down only databases *sid1* and *sid2*.

The exception to this statement is when you do not want to use one of the scripts *on any database*. For example, you can specify that **dbshut** will shut down databases *sid1*, *sid2*, and *sid3*, but **dbstart** will not be used at all.

- To use either script automatically, you will have them called from the startup and shutdown files, as directed in the next sections. In general, put the call to **dbstart** *at the end* of the startup procedures so that other, more basic startup procedures (such as mounting the file system) are performed *before* starting up the database(s).

Put **dbshut** *at the beginning* of the shutdown procedures so that more basic shutdown procedures (such as dismounting the file system) are performed *after* bringing down the database(s).

dbstart

This section tells you how to set up **dbstart** to run automatically.

In the following procedures, the phrase “startup file” refers to the `/etc/rc.local` file.

To set up the **dbstart** script to be called at system startup, perform the following steps:

1. Log in as root.
2. Edit your `/etc/oratab` file.

Find the entries for all the databases that you want to be started up. They are identified by their `ORACLE_SID`s in the first fields. Change the last fields to Y.

For a description of the `/etc/oratab` file, see the “SQL*Net V1 Overview” chapter in the *TARG*.

3. Add a line like the following to the *end* of the startup file (be sure to give the full path of the **dbstart** utility):

```
su - oracle -c /usr/oracle/bin/dbstart &
```

dbshut

On this system, it is impossible to provide automatic database shutdown at system shutdown time. Your system’s **shutdown** command does not read a shutdown script, and therefore **dbshut** cannot be called automatically. Thus, you have to shut down the databases manually (using **dbshut** if desired) before powering off your system.

Note: If your system suffers a power outage, or you shut down the system without shutting down the instance, restarting the instance again after powering back on will fail. This will happen whether or not you have automated the startup. In this case, you must delete the SGA files in `$ORACLE_HOME/dbs`, and then restart the instance.

.sql Scripts

The following .sql scripts are run automatically by the Installer:

- **cat712.sql** for upgrades from Release 7.0.11 to 7.0.12
- **cat711.sql** for upgrades
- **catalog.sql** for installations and upgrades
- **catexp.sql** for installations and upgrades
- **catldr.sql** for installations and upgrades
- **standard.sql** for installations and upgrades with the procedural extension
- **pipidl.sql** for installations and upgrades with the procedural extension
- **pidian.sql** for installations and upgrades with the procedural extension
- **diutil.sql** for installations and upgrades with the procedural extension
- **pistub.sql** for installations and upgrades with the procedural extension

For a complete list of .sql scripts available, see the **\$ORACLE_HOME/rdbms/doc/README.doc** file. All .sql scripts should be run as the **sys** user.

4

UPGRADING AND MIGRATING YOUR ORACLE SYSTEM

This chapter explains how to migrate your ORACLE system from versions of the ORACLE RDBMS prior to Release 7, and how to upgrade your ORACLE7 Server system to later releases. It is organized as follows:

- upgrade and migration paths
- deinstalling old distributions
- migrating from ORACLE RDBMS V6 to ORACLE7
- upgrading from earlier releases of ORACLE7
- import and export utilities
- performing an export upgrade

Upgrade and Migration Paths

This section outlines the upgrade or migration path you must take, depending on the current status of previous ORACLE versions on your system.

Note: The phrase “upgrading the ORACLE7 Server” refers to installing a version of the ORACLE7 Server on a system on which an older ORACLE7 release has been previously installed. If your earlier version is ORACLE RDBS V6.0, this is called a “migration.”

If you are installing ORACLE7 Server for the first time:

- follow the instructions in the “Installing Oracle Products” chapter

If you are migrating from Version 5.1:

- follow the instructions in the “Migrating from ORACLE V5.1 to V6.0” appendix in the *Version 6 Installation and User’s Guide* first, then use the
- *ORACLE7 Server Migration Guide*, with the
- “Migrating from ORACLE V6 to ORACLE7” section of the present chapter

If you are migrating from Version 6.0.26 or earlier versions of 6.0:

- follow the instructions in the “Performing an Export Upgrade” section of the present chapter

If you are migrating from Version 6.0.27 or later versions of 6.0:

- follow the instructions in the *ORACLE7 Server Migration Guide*, with the
- “Migrating from ORACLE V6 to ORACLE7” section of the present chapter

If you are upgrading from ORACLE7 Server Release 7.0.9:

- follow the instructions in the “Deinstalling Old Distributions” section in the present chapter to deinstall your 7.0.10 distribution, and then
- follow the instructions in the section “Upgrading the ORACLE7 Server: Upgrading with the Installer” in the present chapter
- follow the instructions in the “Performing an Export Upgrade” section of the present chapter

If you are upgrading from ORACLE7 Server Release 7.0.10:

- follow the instructions in the “Deinstalling Old Distributions” section in the present chapter to deinstall your 7.0.10 distribution
- follow the instructions in the section “Upgrading the ORACLE7 Server: Upgrading with the Installer” in the present chapter

If you are upgrading from ORACLE7 Server Release 7.0.11:

- follow the instructions in the section “Upgrading the ORACLE7 Server: Upgrading with the Installer” in the present chapter

Deinstalling Old Distributions

When you are upgrading from an older version of ORACLE to the current one, the old distribution must be removed. (The term "Oracle distribution" means all the files and directories unique to a single version release of ORACLE Server and associated Oracle products.)

- If you used the Installer to install the old distribution, then that distribution will be automatically removed when you use the Installer again to upgrade to the current release.
- However, if you did not use the Installer to install the old distribution, you must remove the old distribution manually or with the **deinstall** utility.

Remove old files manually when:

- migrating from any version of ORACLE prior to V6.0.27 to Release 7.0.9 or later
- upgrading from ORACLE Release 7.0.7 to Release 7.0.9 or later

Use the **deinstall** utility when:

- migrating from V6.0.27 or later releases of V6.0 to ORACLE Server Release 7.0.9 or later

Manual Deinstall

When performing an export upgrade, you must manually remove files from outdated distributions. The general procedure is as follows:

1. **Export the data.**
2. **Manually remove distribution files from each directory.**
3. **Install the new distribution.**
4. **Import the data.**

For more details, refer to the section titled "Upgrading ORACLE7 Server" later in this chapter.

The deinstall Utility

Use the **deinstall** script to remove earlier distributions of the products you will upgrade. This script prepares the environment for the upgrade by deleting obsolete files which might not be written over because they were either unwritable or previously located under different directories.

You can also use **deinstall** at installation time to deinstall any products you have read in from the diskette or tape, but do not want to install.

Note: If you deinstall a precompiler (such as Pro*FORTRAN), you must reinstall any other precompiler products you are using on the system in order for the precompilers to function properly.

What deinstall does

The **deinstall** script deletes only the files that were in the original distribution. It does not delete the directory structure, nor files that you have customized, nor files that were generated during installs, such as log files. It does not delete information from the database for installed products.

Syntax of deinstall

The syntax for running the **deinstall** script is:

```
deinstall [-n] [-\?] [all] [product [product] . . .]
```

where:

-n	requests that no confirmation questions be asked about individual products
-\?	displays the deinstall syntax
all	requests that <i>all</i> products be removed
<i>product</i>	specifies the product to be removed. The value for this variable is the directory name in \$ORACLE_HOME .

For example, to deinstall SQL*Plus and SQL*ReportWriter, enter:

```
$ deinstall sqlplus sqlreport
```

To deinstall an entire Oracle distribution with confirmation questions included, enter:

```
$ deinstall all
```

As it runs, the **deinstall** script displays screen messages that report on its progress.

You cannot deinstall the following products without also deinstalling ORACLE. Because these products involve modifications of ORACLE, they cannot be individually deinstalled.

- PL/SQL
- any SQL*Net driver

Note: The **deinstall** utility was first released on ORACLE RDBMS V6.0 with V6.0.27, and first released on the ORACLE7 Server with Release 7.0.9. If it does not yet exist on your system, read it off the first archive of the distribution tape, or from the first distribution diskette, before you begin the following instructions.

To remove an old distribution of one or more Oracle products and then upgrade them to a new version, follow these steps:

- 1. Read any product-specific instructions for upgrading.**

Information for ORACLE appears in the "Upgrading ORACLE7 Server" section in this chapter; information for individual products appears in the "Preparing to Install Oracle Products" chapter, or in the corresponding product chapter in the *Oracle Tools for UNIX Administrator's Reference Guide*.

- 2. Back up any customized files.**

These may include modified makefiles, customized terminal definitions and .r files, or printer resources under the /admin directories of the various products. Products for which this applies include SQL*Forms V3.0, SQL*Menu V5.0, Oracle*Terminal, SQL*ReportWriter, and Oracle*Mail. Also back up any directories or files listed in the product installation instructions for the current version.

- 3. Before running deinstall, make sure any products you want to remove are not in use.**

In general, you should bring the database down when running **deinstall**. If you are upgrading the database, you **must** bring it down. If you are upgrading products, make sure they are not being used. If you are upgrading SQL*Net products, make sure their servers are down.

4. Run the deinstall script to remove the old distribution.

For example, to deinstall the entire distribution, enter:

```
$ cd $ORACLE_HOME/install  
$ deinstall
```

You can now install the new distribution. See the “Installing Oracle Products” chapter.

Migrating from ORACLE V6 to ORACLE7

If you are migrating from ORACLE RDBMS Version 6.0.26 or later, you can use the Installer to perform many of your migration tasks. Choose the option V6 to ORACLE7 Migrate in order to migrate from V6.0. You should be familiar with the information in the *ORACLE Version 6 to Version 7 Migration Guide*.

Migration Stages

Migration from ORACLE Version 6 to ORACLE7 Server consists of three main stages:

1. Install and run the migrate utility against your ORACLE RDBMS V6 database(s).
2. Deinstall the ORACLE RDBMS V6 distribution.
3. Upgrade ORACLE7 Server products.

Note: Once you migrate a Version 6 database to ORACLE Server Release 7, you cannot run a Version 6 RDBMS against the migrated database. Also note that as of Release 7.0.9, you can use the *migrate* utility if you have offline tablespaces.

Procedure

To perform the migration, you will follow the general procedure described in the "Migration Overview" chapter of the *ORACLE7 Server Migration Guide*, with certain modifications for your SunOS system.

Note: This procedure assumes that you have only one `ORACLE_HOME` directory. You must repeat this procedure for each `ORACLE_HOME` you wish to migrate.

Begin by performing Steps 1 through 7, and Step 9, described on pages 1-2 and 1-3 of the *ORACLE7 Server Migration Guide*, with the following caveat:

CAUTION: Step 4 in the *Migration Guide* may not be necessary or feasible (for reasons of time or disk space) on your system. However, if you do choose to rehearse the migration, you must do it *not* on your active `ORACLE_HOME` directory, but on a full test `ORACLE_HOME`, with a test database. (Once you start migrating, you must carry it out on *all* databases on a given `ORACLE_HOME` directory.)

Now perform the following procedure *in place of Step 10 in the ORACLE7 Server Migration Guide*. (Step 10 explains the procedures that the Installer carries out automatically for you.)

1. Shut down and back up all database instances.
2. Install the migration utility on your Version 6 ORACLE_HOME directory.

From the Install Actions Menu of the Installer, choose the V6 to ORACLE7 Migrate option; this will bring you to the V6 to ORACLE7 Migration Action Menu. From this menu, select the Install the Migrate Utility option.

At the Available Products window, select *only* the V6 to ORACLE7 Migration utility to install.

3. Run the migrate utility against your Version 6 databases.

Re-start the Installer, and select V6 to ORACLE7 Migrate from the Install Actions Menu. From the Migrate Action list, select the Run Migrate Utility option. The Installer runs the **migrate** utility against your Version 6 database. This takes some time. When the process is complete, the Installer informs you of the remaining steps you must perform.

Perform this step for all your Version 6 databases, supplying the appropriate *sid* each time you re-start the Installer.

Once the **migrate** utility has been run successfully against your Version 6 database(s) and you have read the remaining information prompts, the Installer returns you to the Installable Products window. You can either call up an operating system shell or exit from the Installer in order to perform the following steps.

4. Modify your **init.ora** file for each of the databases you are migrating.

Exit the Installer and edit your **init.ora** file to remove any obsolete parameters. Obsolete parameters are listed in **\$ORACLE_HOME/rdbms/doc/README.doc**, with the exception of **db_block_write_batch** and **ddl_locks**, which must also be removed.

At this time, you should also add any new **init.ora** parameters listed in the **README.doc** file, including **shared_pool_size**.

5. Rename or remove all control files for each of the databases you are migrating.

For example, you can rename the file `cntrl1sid.dbf` to `cntrl1sid.o`. You *must* perform this step, or the procedure will later fail.

6. Deinstall your Version 6 distribution.

Use the `deinstall` utility to remove your Version 6 distribution. For instructions on using the `deinstall` utility, see the “Deinstalling Old Distributions” section earlier in this chapter.

Note that files generated by the old installation scripts are not removed when you run `deinstall`. Files such as `ksms.o`, or `config.o` should be manually removed from `ORACLE_HOME/rdbms/lib`. You can also remove similar files from the `product/lib` directories, but this is not mandatory.

7. Install Oracle products.

Once you have deinstalled your Version 6 distribution, start a new Installer session (see “Starting the Installer” section of the “Installing Oracle Products” chapter), or return to the one you have shelled out of. Select the V6 to ORACLE7 Migrate option from the Install Actions menu.

From the Migrate Actions list, select the Software Installation option. When you reach the Available Products window, select all the products you want to upgrade for use with Release 7, or install for the first time. *Do not* select the V6 to ORACLE7 Migration Utility from the Available Product list.

During the software installation, the Installer displays status prompts to let you know the current stage of the migration. Pay attention to these prompts so that your support representative can better assist you if problems develop.

8. Upgrade each of the migrated databases to ORACLE7.

Start a new Installer session for each additional database you are migrating. Select V6 to ORACLE7 Migrate from the Install Action list, then select DB Migration/Upgrade from the Migrate Actions list. At the Available Products window, pick all products you wish to upgrade for that database, including ORACLE7 Server. Again *do not* select the V6 to ORACLE7 Migration Utility. Repeat this step for each additional *sid* as necessary.

Now complete the migration by performing Steps 11 through 16 on pages 1–3 and 1–4 of the *ORACLE7 Server Migration Guide*.

Upgrading the ORACLE7 Server

The phrase "upgrading ORACLE7 Server" refers to installing a version of ORACLE7 Server on a system on which an older 7.0 Release has been previously installed. If your earlier version is V6.0, this is called a "migration." If you are migrating from V6.0.27 or later V6.0 releases, refer to the *ORACLE7 Server Migration Guide*, using the "V6 to ORACLE7 Migrate" section of the present chapter for information specific to UNIX systems.

When to install Oracle patches

Your Oracle distribution may include a "patch" tape or diskette, which you read in and install *after* installing the ORACLE7 Server and Oracle products. Use the Installer to apply patches.

If you are upgrading your operating system

Certain operating system upgrades will not require any changes to an installed ORACLE system, depending on whether or not the O/S vendor preserved upward compatibility. Other O/S upgrades will require you to obtain an entirely new Oracle distribution, compatible with the new O/S version.

How to upgrade ORACLE7 Server

Methods for upgrading ORACLE7 Server include the following:

- | | |
|-------------------------|--|
| Oracle Installer | You can use the Installer to upgrade databases that were initially installed using the Installer. |
| export upgrade | This method consists of exporting the database, installing the new version of ORACLE7 Server using the Installer, then importing the old database. See "Performing an Export Upgrade" later in this chapter. |

See the "Upgrade and Migration Paths" section at the beginning of this chapter to learn which versions of ORACLE7 Server on your system require an export upgrade, and which require the **migrate** utility.

Upgrading with the Installer

The Installer option Upgrade An Existing Database allows you to upgrade existing Release 7 ORACLE databases to the current release *if those databases were initially installed with the Installer*.

Before upgrading, you must first install the new software with the Install Software Only options.

Due to changes in the log format, you *must* do a clean shutdown of your current database before upgrading.

If you are using the Installer for the first time you should first deinstall the existing ORACLE distribution using the **deinstall** utility described earlier in this chapter. You can then run the Installer using the Upgrade Existing Database Objects option to install the new ORACLE software and to upgrade an existing database instance.

Import and Export

The **export** and **import** utilities are used for several purposes:

- to back up database files
- to enhance data integrity
- to enable you to migrate from ORACLE RDBMS V5.1 to ORACLE7 Server
- to enable you to migrate from ORACLE RDBMS V6.0 releases prior to V6.0.27 to ORACLE7 Server
- to enable you to migrate from ORACLE7 Server Release 7.0.7 to Release 7.0.9

When migrating from versions prior to 6.0.27 of the ORACLE RDBMS to ORACLE Server Release 7.0.9 or later, you must perform a full export and import and run the automated installation procedure again. You must upgrade all of your Oracle products when you perform this upgrade of ORACLE.

A full export and import of your data is necessitated by changes in the log files. The full reinstall is required because of the directory organization changes. For the same reasons, you must fully reinstall all Oracle products you wish to run with the ORACLE Server Release 7.0.9, or later releases.

For a complete discussion of the **export** and **import** utilities, refer to the *ORACLE7 Server Utilities User's Guide*.

Import and Export with Multiple Tapes

Under the ORACLE Server Release 7.0, the **import** and **export** utilities handle multiple tape files. If the filename entered is a tape device, you will be prompted to specify the volume size of each tape. When the end of each volume is reached, you will be prompted to mount a new volume and press [RETURN] when the tape is ready.

When volume sizes are specified, a number of bytes is expected. A number may end with **m**, **k**, or **b** (or **M**, **K**, or **B**) to specify multiplication by 1024*1024, 1024, or 1, respectively. You may also specify the volume size by entering **volsize=size** on the command line, or by putting the parameter in a file (**PARFILE**, specified by entering **PARFILE=file**). You cannot specify fractions (such as 1.2M). Here is an example of mounting multiple tapes:

```
$ imp file=/dev/rmt0 volsize=100M
```

```
...
```

Please mount the next volume, and hit <ret> when you are done

```
...  
Please mount the next volume, and hit <ret> when you are done  
...  
Import terminated successfully
```

```
$ exp
```

```
...  
Export file: expdat.dmp > /dev/rmt/0  
Volume size (<ret> for no restriction) > 950K  
...  
Please mount the next volume, and hit <ret> when you are done  
...  
Please mount the next volume, and hit <ret> when you are done  
...  
Export terminated successfully
```

If you create the database manually with the **create database** statement, you must use SQL*Plus and start **\$ORACLE_HOME/rdbms/admin/expview.sql** before you can import. When using Oracle Installer, this is done for you.

Performing an Export Upgrade

An export upgrade is always an option, whether you are migrating to ORACLE7 from Version 6 of the ORACLE RDBMS, or upgrading from an earlier release of the ORACLE7 Server.

Preliminary Operations

This section explains the tasks you must perform before you can carry out an export upgrade of ORACLE7 Server.

Plan your installation

Be sure to read the section "Planning Your Installation" in the "Preparing to Install Oracle Products" chapter for a description of essential steps to perform before installing ORACLE7 Server.

Save customized files

Before upgrading to ORACLE7, save any files you may have customized from the old version. This applies to modified makefiles, customized terminal and printer definitions for SQL*ReportWriter or Oracle*Mail, or Oracle*Terminal files, and any applications you may have built in the ORACLE home directory.

In each product chapter of the *Oracle Tools for UNIX Administrator's Reference Guide*, there is a section at the beginning that lists any special product files you should save before deinstalling.

Because the install script will replace each product's customized files, save them outside of the \$ORACLE_HOME hierarchy before you run the script. You do not have to save the *oracle* account's *.login*, *.cshrc* or *.profile* file, because the script renames them to *.login.OLD*, *.cshrc.OLD* and *.profile.OLD*, respectively.

Remove old products

You can use the **deinstall** script to remove old product distributions. This utility was released in V6 with V6.0.27, and in ORACLE7 with Release 7.0.9. If you are upgrading from a version that did not have **deinstall**, you must first read in the **deinstall** script from the distribution medium. See the section on "Deinstalling Old Distributions" earlier in this chapter for guidelines on using the **deinstall** script.

Set up certain user accounts

If you intend to install either SQL*Net Async or SQL*Net DECnet for the first time, set up their user accounts (*sqlnet* or *dni*, respectively) in */etc/passwd* and */etc/group*, along with their corresponding home directories.

Perform the Export Upgrade

If you are migrating from ORACLE RDBMS Version 6.0.26 or an earlier version of 6.0 perform the following steps to upgrade to the current ORACLE7 Server release.

Note: This procedure assumes that you have only one **ORACLE_HOME** directory. You must repeat this procedure for each **ORACLE_HOME** directory you wish to upgrade.

If you are upgrading from Release 7.0.9 to the current release of ORACLE7 Server, you must perform an export upgrade as follows:

To perform the migration, you will follow the general procedure described in the "Migration Overview" chapter of the *ORACLE7 Server Migration Guide*, with certain modifications for your SunOS system.

Begin by performing Steps 1, 2, 3, 8, and 9 described on pages 1–2 and 1–3 of the *ORACLE7 Server Migration Guide*.

Now perform the following procedure *in place of* Step 10 in the *ORACLE7 Server Migration Guide*. (Step 10 explains the procedures that the Installer carries out automatically for you.)

1. **Shut down the running ORACLE instance and restart it in DBA mode:**

```
SQLDBA> connect internal
SQLDBA> shutdown
SQLDBA> startup dba
```

2. **Export the tables you have created.**

Refer to the description of export and import in this chapter.

3. **Shut down the instance.**

4. **Back up the Oracle distribution and the control files, database files, redo log files, and init.ora file.**

WARNING: If the database is not backed up correctly and the upgrade fails, you will not be able to switch back to the earlier version.

5. **Remove the distribution and database files to reclaim disk space.**

Do not remove any files you have customized.

6. **Check that you have enough disk space on the target disk.**

Refer to the disk and memory requirement charts in the "Key to ORACLE on SunOS" chapter for more information.

7. **Select the Fresh Install option from the Install Actions Menu**

Follow the instructions in the "Installing Oracle Products" chapter for Software/Database Fresh Installation.

8. **Import the export files created in Step 2.**

Now complete the migration by performing Steps 11 through 16 on pages 1–3 and 1–4 of the *ORACLE7 Server Migration Guide*.

CHAPTER

5

CONFIGURING YOUR ORACLE SYSTEM

This chapter presents an overall approach to ORACLE system configuration. In addition, it describes certain other configuration options.

The chapter is organized as follows:

- recommendations for configuring your database
- options for customizing your installation
- relinking
- installing ORACLE in a network configuration

Recommendations for Configuring Your Database

This section is a continuation of the recommendations presented in the "Preparing to Install Oracle Products" chapter, and contains suggestions for creating and administering databases. These steps are not mandatory, but may provide helpful information for configuring your databases whether or not you have followed the steps for the Oracle Base Structure in the "Preparing to Install Oracle Products" chapter. The following is a brief list of the recommended steps detailed in this section:

Steps for database administration:

1. Assign database names.
2. Name database instances.
3. Create administration subdirectories.

Steps for instance parameter files:

4. Create config.ora file.
5. Set config.ora parameters.
6. Create init.ora files.

Steps for ORACLE data:

7. Create ORACLE database data directories.
8. Set control_files parameter.
9. Create redo log files.
10. Create data files.

Steps for tablespaces and segments:

11. Create tablespaces.
12. Separate database segments into tablespaces.
13. Administer tablespaces.
14. Create rollback segments.

Database Administration

Assign database names

Give each ORACLE database a connotative name, unique within four characters, case-insensitive, by specifying a value for the **db_name** instance parameter.

Note: Not all UNIX operating systems are restricted by four character file names. This recommendation is based on the concept of portability between systems as well as economy of usage.

Selection of a database name is an important first step in the administration of a database. An ORACLE database name must uniquely identify the database from any other ORACLE database on the file system.

Since the ORACLE database name will become a path component of each database file, it is critical to give different names to any two databases that might have files on the same disk drive.

Choose a name that is unique even under case-insensitive inspection. As you will see in discussions of the heterogeneous distributed server capability of ORACLE7 Server, some operating system command line interpreters are case-insensitive.

The four-character name length limitation makes it especially important to maximize the connotative value of the characters you use. Avoid strings like "database" or "db" in the ORACLE database name. Embedding information about an object's type into its name does nothing more than count against the name's character limit.

Name database instances

Name each ORACLE instance with the same name as that of the database it manipulates, four characters maximum (for non-parallel server databases) by setting a value for **ORACLE_SID**.

This step minimizes confusion in determining the relationships among multiple instances and multiple databases on the same machine.

An ORACLE instance is a collection of processes and their associated memory structures. The name associated with the ORACLE instance processes is known as the ORACLE system identifier (*sid*). ORACLE uses the *sid* stored in the UNIX environment variable **ORACLE_SID**.

The DBA must set **ORACLE_SID** before starting an instance, and the *sid* becomes embedded in the **ps(1)** command text for each ORACLE process. The DBA can change the *sid* each time the instance is restarted, but changing an instance name can confuse users.

Users choose the instance (and hence the database) they want to use by selecting among the instance names available. The master DBA should record available instance names for a UNIX system in `/etc/oratab` or `/etc/listener.ora`. Since users in sophisticated systems can come into daily contact with the ORACLE instance name, it is important that the name of an instance be chosen carefully.

The ORACLE instance name length should be held to a maximum of four characters because the value of `ORACLE_SID` becomes a substring of several operating system file and process names. Using ORACLE instance names of no more than four characters ensures that you have the greatest possible flexibility when selecting connection strings (which include node names and *sids*) to link together an optimized ORACLE7 Server distributed server environment.

Making the instance name a different case than the database name can serve to clarify the distinction between the instance name and the database name. Lower case is chosen for the instance name because the instance name is typed more frequently, and lower case is easier to type. Users selecting from among several instances type instance names daily. Only the DBA comes into contact with the more cumbersome uppercase database name.

Picking an uppercase ORACLE instance name on UNIX can ease the burden of mapping instance names across networks of machines, some of which may not honor case sensitivity (DEC VAX/VMS is a good example of an operating system with a case-insensitive command line interpreter).

The cost of choosing a lowercase instance name on a UNIX system is that DEC VAX/VMS users who wish to connect to the UNIX system database will be required to explicitly quote all references to the lowercase UNIX *sid*.

Create database administration subdirectories

Administrative files, such as database creation scripts, trace files, logbook entries, and database exports, are associated with both the database data and the instance used to run them. Naturally it is necessary to separate files of one database from similar files of another database.

By placing the administration directory that will contain all administrative files associated with a database outside of the ORACLE software version subtree, the DBA does not need to pay special attention to these files at software upgrade time.

For example, upgrading the *d1* database from ORACLE7 Server Release 7.0.12 to 7.0.13 requires a switch of software, but the move to a new version of ORACLE software should not require maintenance of, for

example, the SQL programs, trace files, and archived log files associated with *d1*.

For each database whose **db_name=d**, create the following directories in **\$ORACLE_BASE/admin/d**:

```
adhoc arch bdump cdump create exp logbook pfile udump
```

Give these directories owner *oracle*, group owner *dba*, and mode 775.

These directories will be used in the course of creating and administering the ORACLE database, where:

adhoc	stores ad hoc SQL scripts that apply only to a single given database
arch	stores archived redo log files
bdump	stores ORACLE background process trace files
cdump	stores ORACLE core dump files
create	stores programs used to create the database
exp	stores database export files
logbook	stores output from DBA programs run in the database
pfile	stores instance parameter files such as init.ora and config.ora
udump	stores ORACLE user SQL trace files

Giving these directories mode 775 will permit any member of the *dba* group to read, write, and execute files there. This is necessary so that DBAs can make administrative files (like logbook entries), analyze and remove trace files, and make database exports. Other users are permitted to view these administrative files.

Instance Parameter Files

ORACLE instance parameters, often called **init.ora** parameters, determine the character of an ORACLE instance and its connection to an ORACLE database. Instance parameters are specified in the **init.ora** file and can be divided into two groups:

- configuration parameters
- tuning parameters

Configuration parameters define the relationship between the database and its environment and are contained in the **config.ora** file. Tuning parameters determine the variable characteristics of an ORACLE instance and are contained in the **init.ora** file. Because they are subject to a dramatically different pace of modification, **config.ora** and **init.ora** files are stored in different UNIX file system subtrees.

Create a **config.ora** file

For each ORACLE database named *d*, create a default configuration file named **\$ORACLE_BASE/admin/d/pfile/config.ora**. Give the file owner *oracle*, group owner *dba*, and mode **660**. Include the settings of this file in the **init.ora** file indirectly through the **ifile** instance parameter.

By placing database configuration parameters in the database administration directory, you exploit the natural separation of parameters closely associated with the database (which rarely change) from parameters closely associated with the instance processes and memory structures (which change frequently), such as during tuning and software upgrades.

DBAs typically have many **init.ora** files that share configuration parameter settings. The same logic that inspires software engineers to include C language header files motivates us to isolate configuration parameters in an included file. Using a single **config.ora** file that is included by multiple **init.ora** files ensures that only a single modification point is required to simultaneously update the behavior of several parameter files.

The **config.ora** file resides in the administration directory for a specific database, outside of any database software subtree. Because each **init.ora** file contains the full pathname of a **config.ora** file, **config.ora** is given permanent residence in a directory that remains constant through the database upgrade process. (Remember that the value of **ORACLE_HOME** changes with every database upgrade.) If a **config.ora** file were kept in the **\$ORACLE_HOME** subtree, an administrator would be required to update every **ifile** parameter value at every upgrade.

Set **config.ora** parameters

Set the following **config.ora** parameters for each instance accessing the database whose **db_name=d**:

- **control_files** set to a list of file names
- **init_sql_files** set to the list of SQL programs that should be executed at database creation time
- **db_block_size** set to an appropriate integer value
- **db_name** set to a database name
- **background_dump_dest** = \$ORACLE_BASE/admin/d/bdump
- **core_dump_dest** = \$ORACLE_BASE/admin/d/cdump
- **log_archive_dest** = \$ORACLE_BASE/admin/d/arch/archn.log
- **user_dump_dest** = \$ORACLE_BASE/admin/d/udump

Setting these values greatly facilitates administrative operations upon the various operating system files an ORACLE instance creates. For example, separating user process trace files from background process trace files simplifies the task of determining which file relates to a particular SQL trace analysis. Antiquated background trace files in their own directory can be easily removed without fear of destroying a user's precious SQL trace histories.

Control files store information that allows the instance to compute the names of the operating system files it is supposed to manipulate.

An ORACLE database block size can be 512, 1024, 2048, 4096, or 8192 bytes. In general, larger block sizes tend to yield more efficient database read cache hit ratios. The cache hit ratio is the proportion of read requests that can be serviced by previously executed reads into the System Global Area (SGA).

The cache hit ratio is calculated as $(r-p)/r = 1-p/r$, where r is the number of read requests, and p is the number of actual physical reads from the database. Larger blocks contain more rows than smaller blocks. So, if the number of database blocks held in the SGA remains fixed, the chances of a desired row's presence in the SGA are better if the individual blocks are bigger.

However, larger blocks incur cost because more memory is required to store an SGA holding large blocks than is required to hold an equal number of smaller blocks. More disk space is required to store minuscule database segments consisting only of one header block and one storage block.

Most UNIX sites use database block sizes of 2048 or 4096. A 2048-byte block strikes a good balance between performance and storage. Larger block sizes are recommended only for sites with ample system memory and the sophistication to understand the resulting inaccuracies in sizing estimates provided by their application software vendors. For more information about recommended block size for your operating system, see the "Key to ORACLE" chapter.

Setting the archive log file destination to **arch.log** in the **arch** administration directory will result in log archive files with base names of **arch*n*.log**, where *n* is the log sequence number of the archive.

Setting the background dump destination to the **bdump** administration directory causes ORACLE to write its background process diagnostics to files matching the pattern **opid_upid.trc** in this directory, where *opid* is the ORACLE process identifier and *upid* is the UNIX process identifier of the background process.

Setting the user dump destination to the **udump** administration directory causes database sessions using the **SQL trace** facility to write trace files there. Users who need to read trace files created in the user dump directory are permitted to do so, but file deletion is coordinated through a member of the **dba** group.

The following is a sample **config.ora** file:

```
$ cat $ORACLE_BASE/admin/d1/pfile/config.ora
# @(#)config.ora          1.1 - database configuration parameters
#
# Bob Smith
# 92/02/23
control_files             = (/u01/ORACLE/d1/control.ctl,
                             /u03/ORACLE/d1/control.ctl,
                             /u06/ORACLE/d1/control.ctl)
init_sql_files            = (?/dba/sql.bsq,
                             ?/rdbms/admin/catalog.sql,
                             ?/rdbms/admin/expview.sql)
db_block_size             = 2048
db_name                   = d1
background_dump_dest      = $ORACLE_BASE/admin/d1/bdump
core_dump_dest            = $ORACLE_BASE/admin/d1/cdump
log_archive_dest          = $ORACLE_BASE/admin/d1/arch/arch.log
user_dump_dest            = $ORACLE_BASE/admin/d1/udump
```

Create **init.ora** files

One of the first actions taken by the **SQL*DBA startup** command is the opening of the **init.ora** file. This file determines the configurable attributes of the instance processes and memory structures.

For each ORACLE instance *sid*, create the following **init.ora** files:

- **\$ORACLE_BASE/admin/d/pfile/init.ora**: the default **init.ora** file with production rollback segment activation and all tuning parameters
- **\$ORACLE_BASE/admin/d/pfile/init_0.ora**: the **init.ora** file containing no rollback segment activation and no tuning parameters
- **\$ORACLE_BASE/admin/d/pfile/init_1.ora**: the **init.ora** file containing maintenance rollback segment activation and no tuning parameters. Give the files owner *oracle*, group owner *dba*, and mode **660**.

Sample contents of the three **init.ora** files are shown below. Since the **_0** and **_1** files are used only in periods of rollback segment maintenance when performance is not an issue, there is no need to maintain tuning parameters in all three files. The strategy of recording tuning parameters only in the production **init.ora** file relieves the database administrator of maintaining parallel updates in more than one file. Using the **ifile** text inclusion parameter with the **config.ora** file eliminates the need to coordinate replicas of the same information among three different files.

Note that **ifile** requires specification of the included file's full pathname. No environment variables or relative pathnames are allowed. This technical restriction is of little consequence because the pathname of the **config.ora** file never changes, not even during a database upgrade.

The **init.ora** file used for production instances is given the default name expected by the **SQL*DBA startup** command. Using the default name relieves the administrator of remembering to specify the value of **pfile** at every instance startup. Only when rollback segment maintenance is necessary will the DBA be required to specify an **init.ora** file name. Maintenance **init.ora** file names whose basenames end in **_0** and **_1** clearly connote the number of rollback segments activated by each file.

A secondary benefit of restricting the *sid* to four characters is that some implementations of UNIX limit file name lengths to fourteen characters. The recommended **init.ora** file naming convention of appending **_0** or **_1** to the basename meets the fourteen-character restriction for even the longest instance name recommended.

Sample 1: The following output is a sample instance parameter file excerpt, `init.ora`:

```
$ cat $ORACLE_BASE/admin/d/pfile/init.ora
# @(#)init.ora      1.1 - instance parameters, production rbs
#
# Bob Smith, Oracle USA
# 92/02/23
# include database configuration parameters
ifile          = /u01/home/dba/oracle/admin/d1/pfile/config.ora
# dictionary cache
# program global area
sort_area_size      = 262144 # max sort memory size
open_cursors        = 255 # n(cursors per process)
# system global area
db_block_buffers     = 1000 # n(blocks in SGA db buffer)
db_file_multiblock_read_count = 8 # n(blocks read per i/o by fts)
# locking
enqueue_resources   = 1220 # <= (ddl_locks + dml_locks + 20)
dml_locks           = 200 # n(xactns) * n(modified tables)
# redo logging
log_buffer           = 65536 # 64Kb: medium-sized
log_checkpoint_interval = 4096 # size of log file in UNIX
                        # blocks
log_files            = 32      # will reduce SGA size in ORACLE7
# sequences
sequence_cache_entries = 30 # n(seq nums cached)
sequence_cache_hash_buckets = 23 # < sequence_cache_entries
processes            = 20 # n(connections)
rollback_segments    = (r01,r02,r03,r04)
```

Sample 2: The following output is a sample instance parameter file, `init_0.ora`:

```
$ cat $ORACLE_BASE/admin/d/pfile/init.ora
# @(#)init.ora      1.1 - instance parameters, no rbs
#
# Bob Smith, Oracle USA
# 92/02/23
# include database configuration parameters
ifile          = /u01/home/dba/oracle/admin/d1/pfile/config.ora
```

Sample 3: The following output is a sample instance parameter file, `init_1.ora`:

```
$ cat $ORACLE_BASE/admin/d/pfile/init_1.ora
# @(#)init_1.ora      1.1 - instance parameters, maintenance rbs
#
# Bob Smith, Oracle USA # 92/02/23
# include database configuration parameters
ifile      = /u01/home/dba/oracle/admin/d/pfile/config.ora
rollback_segments = (r0)
```

ORACLE Database Data

This section explains certain precautions you should take to protect your data, and describes steps you should take to best manage your data.

Data storage considerations

Database data should not generally be placed in the `dbf` directory because:

- Putting all ORACLE data in a single `dbf` directory unnecessarily restricts the amount of data that can ever reside on the system. A single directory mounted on a single disk section cannot accommodate arbitrarily large amounts of data.
- The `dbf` directory is a subtree of the directory containing the ORACLE software. Placing ORACLE data in `dbf` forces unnecessary effort, risk, and expense in caring for data whenever the ORACLE software is upgraded.
- Placing all database files on a single disk places an unnecessarily high demand on a single disk resource, resulting in a poor performing, I/O bottlenecked database.
- Placing all database files on a single disk is unnecessarily risky because it permits a single drive failure to annihilate an entire database.

The following recommendations offer a solution to the problem of how to store massive amounts of ORACLE data on a UNIX system.

/u01	/u02	/unn
/\$ORACLE_BASE	/\$ORACLE_BASE	/\$ORACLE_BASE
/ORACLE	/ORACLE	/ORACLE
/db_name1	/db_name1	/db_name1
ctrl01.ctl	ctrl01.ctl	ctrl01.ctl
redo01.log	redo01.log	redo01.log
redo02.log	redo02.log	redo02.log
redo03.log	redo03.log	redo03.log
system01.dbf	system01.dbf	system01.dbf
rbs01.dbf	rbs01.dbf	rbs01.dbf
temp01.dbf	temp01.dbf	temp01.dbf
users01.dbf	users01.dbf	users01.dbf
/db_name2	/db_name2	/db_name2

FIGURE 5-1 Recommended placement of ORACLE data files

VLDB Considerations

One of the common requirements of very large database (VLDB) sites is that disk drive hardware failure must impact as small a number of production systems as possible. To achieve this goal, VLDB sites typically isolate databases by devoting entire sets of drives to individual projects, cautiously avoiding placement of two different projects on the same drive.

In the VLDB environment, the choice of non-connotative mount point names runs contrary to the goal of the administrator, which is to specify that a given mount point is intended to house only the database files of a single specific ORACLE database. Sites meeting the following VLDB criteria can benefit from mount point and file naming conventions different from those given in the previous sections:

- each database can be given a number of disk drives devoted solely to storage of control, redo log, and data files from that database
- the number of whole disk drives dedicated to each database is sufficiently large that total system I/O can be balanced effectively

For sites meeting these VLDB criteria, name the mount points of disk drives devoted to ORACLE data using the pattern */oradmm*, where *d* is

the **db_name** of the database to which the mount point will be dedicated, and *mm* is a unique two-digit key that distinguishes one mount point from another.

By collapsing the three layers of the recommended ORACLE data file trees, (*unn/ORACLE/d*), the DBA accomplishes the VLDB goal of fault isolation. All control files, redo log files, and datafiles will be located in this directory. See the following sections for naming conventions.

Create database data directories

For each ORACLE database on the system, create a directory on each UNIX mount point matching the pattern **/ORACLE/d*, where *d* is the **db_name** of the database. Give each directory owner *oracle*, group owner *dba*, and mode 750.

This directory separates data in one ORACLE database from data in other ORACLE databases.

Giving these directories mode 750 makes it possible for **dba** group members to inspect ORACLE database files, but only the master DBA, logged in as *oracle*, will be able to move or delete them. This convention forces the coordination of major database surgery through the master DBA.

When you make a special set of directories for a database, never put files for that database anywhere else, and never put anything but files for that particular database there.

Set **control_files** parameter

ORACLE control files contain structural information about the ORACLE database, including relatively static data such as UNIX file names, and more dynamic information like the current redo log sequence number. Control files are created at database creation time and maintained automatically by ORACLE. To activate the creation and automatic maintenance of control files, the database creator needs only to specify all desired control file names in the **control_files** instance parameter before executing the SQL*DBA **create database** statement.

For each database instance list unique names for the **control_files** instance parameter value such that:

- there are at least three control files
- no two control files of a given database are on the same physical drive
- control file names match the pattern **/ORACLE/d/controlnn.ctl*, where *d* is the value of **db_name**, and *nn* is a unique two digit number that distinguishes one control file name from another

Losing your last control file is a calamitous event that requires the DBA to perform database recovery steps. Therefore it is essential that the DBA create and maintain two control files at the very minimum. Having three copies ensures that even with the loss of one control file, there remains a safe duplication.

By using this pattern to name ORACLE control files, the DBA ensures that:

- all control files of a given database are easy to locate
- control files of one ORACLE database are never confused with control files of another ORACLE database
- administrative programs (such as UNIX backup scripts) need not be altered if a control file is moved from one disk to another

Since control files that are maintained by ORACLE are identical copies of a single file, it is unnecessary and in fact misleading to give control files different basenames. The suggestion that no two control files exist in the same directory ensures that each control file has a unique pathname. Note that the database name is a component of the fully qualified data file name and need not be a part of the file's basename.

Create redo log files

ORACLE redo log files record information necessary to re-create the SGA database buffer contents in case of cpu or disk failure. Redo log files are written sequentially and read only during archiving and recovery.

For each ORACLE database, create redo log files such that:

- the redo log files are large enough so there are no checkpoint-induced file busy waits
- there is a sufficient number of redo log files so there are no archiver-induced file busy waits
- all redo log files of a given database are on the same lightly loaded disk drive or distributed in a ping-pong arrangement across more heavily loaded drives
- log files are isolated to the greatest degree possible on hardware serving few or no files that will be active while the database is in use
- redo log file names match the pattern **/ORACLE/d/redonn.log*, where *d* is the value of **db_name**, and *nn* is a unique two-digit number that distinguishes one redo log file name from another

By using this pattern to name ORACLE redo log files, the DBA ensures that:

- all redo log files of a given database are easy to locate
- redo log files of one ORACLE database are never confused with redo log files of another ORACLE database
- administrative programs (such as UNIX backup scripts) need not be altered if a redo log file is moved from one disk to another

Note that the database name is a component of the fully qualified data file name and need not be a part of the file's basename.

It is desirable to isolate ORACLE redo log files from ORACLE data to minimize the possibility that both the database and its recovery mechanism can be lost with a single drive failure.

It can aid performance to separate redo log files from other ORACLE data. Although ORACLE averages less than one redo log file I/O per transaction, sites with extremely high transaction rates may experience high I/O rates to its redo log files. Separating redo logs from other database files can help balance disk I/O load.

A checkpoint-induced "file busy wait" occurs when a redo log file fills before **dbwr** can complete a checkpoint. An archiver-induced file busy wait results when **lgwr** attempts to reopen a redo log file before **arch** has finished archiving it.

File busy waits are reported in the **lgwr** trace file. To eliminate file busy waits, the DBA may either add a third redo log file or replace the existing redo logs with larger files. A reasonable first guess at an appropriate log file size in an interactive system is 128 kilobytes per database connection for systems with three redo log files, or 256 Kb per database connection for systems with two redo log files.

Create data files

ORACLE data files contain all database segments that exist in ORACLE tablespaces. The tablespace is the primary point of contact between ORACLE and its host operating system. Without separating two segments into two different tablespaces, it is difficult to ensure that the two segments will reside on different disks. Once segments have been separated rationally among tablespaces, the following section guides the DBA to placement of the tablespace data files.

For each ORACLE database, create data files such that:

- tablespaces whose segments will compete for disk resource are separated across different disk resources
- data file names match the pattern `/*ORACLE/d/tnn.dbf` where *d* is the value of `db_name`, *t* is the name of the tablespace with which the file is associated (`SYSTEM`, `TOOLS`, `USERS`, etc.), and *nn* is a unique two-digit number that distinguishes one tablespace data file from another

By using this pattern to name ORACLE data files, the DBA ensures that:

- all data files of a given database are easy to locate
- data files of one ORACLE database are never confused with data files of another ORACLE database
- all data files of a given tablespace are easy to locate
- administrative programs (such as UNIX backup scripts) need not be altered if a data file is moved from one disk to another

Naming a data file for the tablespace with which it is associated clearly identifies the data file. Including a unique number string in the data file basename uniquely identifies data files associated with the same tablespace. Making this number string two digits long ensures that data file names will sort properly, even if there happen to be more than ten files associated with a single tablespace. Note that the database name is a component of the fully qualified data file name and need not be a part of the file's basename.

It is not critical to accurately estimate the load balance of a new database as its tablespaces are created. This is fortunate because the unique requirements of individual applications would make it nearly impossible to formulate an accurate estimate. A well planned database configuration requires little tuning, but load balancing a database that has been created using these suggested steps is an easy task whose most time-consuming component is the UNIX `copy` command.

Example: The following output shows a SQL*DBA script that follows the recommendations in this chapter. The *d1* database created here will have three automatically maintained control files, three 5-Mb redo log files, and a 25 Mb `SYSTEM` tablespace. All *d1* database files, have names matching the pattern `/*ORACLE/d1/*`.

```
$cat $ORACLE_BASE/admin/d1/create/1-rdbms.sql
REM @(#)1-rdbms.sql - create d1 database, instance name sid1
REM
REM Bob Smith
REM 91/12/19
...
```

```

REM * Start the sid1 instance (ORACLE_SID here must be set to
sid1).
REM *
startup nomount pfile=${ORACLE_BASE}/admin/d1/pfile/init_0.ora
connect internal
REM * Create the d1 database.
REM *
create database d1
maxdatafiles 64
maxlogfiles 32
datafile
    '/u04/ORACLE/d1/system01.dbf' size 25m
logfile
    '/u01/ORACLE/d1/redo01.log' size 5m,
    '/u01/ORACLE/d1/redo02.log' size 5m,
    '/u01/ORACLE/d1/redo03.log' size 5m;
...
REM * Create a tablespace for rollback segments.
create tablespace rbs datafile
    '/u05/ORACLE/d1/rbs01.dbf' size 25m,
    '/u08/ORACLE/d1/rbs02.dbf' size 25m;
default storage (
    initial 128k
    next 128k
    pctincrease 0
    minextents 20
    maxextents 40
);
...

```

Tablespaces

Tablespace is an ORACLE name for an entity that relates multiple database segments to multiple operating system files. A number of factors affect decisions about separating ORACLE segments into different tablespaces:

- fragmentation character
- I/O distribution
- administrative needs

Each time a database segment is dropped, it causes fragmentation in the segment's tablespace. Tablespace fragmentation can cause inefficient use of freespace, which can lead to application failure. The database designer must carefully separate segments into different tablespaces to isolate the ill effects of tablespace freespace fragmentation.

Only at the tablespace level can the administrator determine what set of operating system files a segment will occupy. Consequently, the

database designer must carefully separate segments into different tablespaces to facilitate I/O load balancing.

Only at the tablespace level can the administrator specify collections of segments for backup or recovery or restrict users' privileges to consume database space. Hence, the database designer must carefully separate segments into different tablespaces to enable the administrator to adequately control segment groups.

Create tablespaces

Every ORACLE database, regardless of the complexity of the applications within it, requires places to store its own data dictionary, rollback segments, temporary segments, database tools, and user application segments (tables, indexes, clusters, etc.). Consequently, even the smallest ORACLE databases should contain separate tablespaces to accommodate these different types of segment.

Create tablespaces such that:

- each ORACLE database has at least the following special tablespaces:
 - SYSTEM**
 - RBS** or **RBS_{nn}** (where *nn* is a unique two-digit integer key)
 - TEMP** or **TEMP_{nn}** (where *nn* is a unique two-digit integer key)
 - TOOLS**
 - USERS**
- each tablespace has a connotative name with a maximum of eight characters

The five required tablespaces contain exactly the following segments:

SYSTEM	should contain SYS-owned dictionary segments only. The only non-dictionary segments permitted to reside in the SYSTEM tablespace are the rollback segments named SYSTEM and R0 .
RBS or RBS_{nn}	should contain rollback segments only. Place all active rollback segments except for the one named SYSTEM in the RBS or an RBS_{nn} tablespace. The rollback segment named SYSTEM , in the SYSTEM tablespace, is created by sql.bsq and, when another rollback segment is present, it is used only by recursive SQL operations.

TEMP or TEMPnn	should contain temporary segments only. Alter each ORACLE userid so that its temporary tablespace is TEMP or a TEMPnn.
TOOLS	should contain SYSTEM -owned segments only. Any segments associated with database tools and traditionally owned by SYSTEM belong here. Examples include segments used by SQL*Plus (help and user profile tables), SQL*Forms, CRT, and SQL*ReportWriter. Naturally, any tool singled out for special development activity or having special growth demands may require its own tablespace.
USERS	should contain users' personal tables and miscellaneous activity. Any ORACLE userid without an immediately obvious default tablespace should have its default tablespace set to USERS .

We do not recommend that you lump dissimilar segments into one large tablespace, especially the **SYSTEM** tablespace.

The five fundamental tablespaces that should be present in every database, regardless of size, have short, connotative names. Additional tablespaces should be given similarly concise names. Never embed a string like "tablespace" or "tbs" in the name of a tablespace. Since a tablespace name is always identifiable as a tablespace name by context, attempting to embed the information "this is a tablespace name" into the name merely counts against the name's length limit. Tablespace name lengths are restricted to eight characters to ensure that a data file's basename does not exceed the UNIX portability limitation of fourteen characters.

Separate segments
into tablespaces

Separate database segments into different tablespaces according to the following segment attributes:

- if segment lifespans or growth characteristics differ, place the segments into separate tablespaces
- if two segments are likely to require disk access at the same time, place the segments into separate tablespaces
- if two segments represent objects from different projects or applications, place the segments into separate tablespaces

Excessive tablespace freespace fragmentation can impede the performance of DDL executed by or on behalf of an application. For example, a **create table** statement can take significantly longer in a

fragmented tablespace than an identical statement in a clean one. Insertion of a row that motivates allocation of a new extent to a segment can also perform worse in a fragmented tablespace than in one that is free of fragmentation.

Tablespace freespace fragmentation can occur only when a user drops a database segment. The ORACLE database administrator can exploit this fact by separating segments among different tablespaces based on the propensity of a segment to fragment its tablespace. The following table lists examples of segments and their fragmentation propensities.

Freespace Fragmentation Propensity	Example Segments
zero	data dictionary segments
low	tools segments static production data static production indexes
moderate	applications interim tables rollback segments
high	development test data development test indexes
very high	temporary segments

It is especially important to separate **SYSTEM** tablespace data dictionary segments from all other segments in the database because **SYSTEM** cannot be "defragmented" by any means other than rebuilding the database.

Because dictionary segments are never dropped, the dictionary has a zero propensity to fragment its tablespace. To guarantee zero **SYSTEM** freespace fragmentation, it is sufficient for the administrator to prohibit creation of all database segments in **SYSTEM** after the **create database** statement has been executed. The administrator may implement this policy by ensuring that no user except for **SYS** has a resource privilege on the **SYSTEM** tablespace, and no user uses **SYSTEM** as its temporary tablespace.

Reclamation advantages exist in separating any two projects that live by different funding schedules, or that consume tablespace at different rates. Consider the effects of storing a small but steadily growing bespoke Helpdesk application in the same tablespace with a massive Oracle General Ledger. If funding for the bespoke project terminated and Helpdesk segments were dropped, then the tablespace storing the huge General Ledger segments would be left with free fragments too small to accommodate an extent from one of the GL tables that grows in increments of 50 Mb.

The ORACLE7 Server allows the administrator to balance I/O load across multiple disk drives in a consistent way, regardless of the operating system. One of the benefits of the ORACLE7 Server is the independence of its administration from that of its host operating system. ORACLE has this independence in part because its thin interface to operating system files is pushed down to the tablespace/data file level. That is, only at the tablespace level is the administrator permitted (or required) to specify what disk a database segment will live on. Consequently, to later separate contending database segments across different disk drives, it is necessary to first place these segments in different tablespaces.

ORACLE7 Server offers several tools for manipulating objects in collections that are partitioned along tablespace boundaries. Database space resources and online backups are administered at the tablespace level, motivating a range of decisions to separate segments into different project-focused tablespaces. Experienced shops separate projects into different tablespaces for a variety of reasons.

Accounting for space used by a project is more efficiently accomplished by tablespace than, for example, by schema (i.e., by ORACLE userid). It is simpler to determine how much space is consumed by a project occupying a single tablespace than to determine how much space is consumed by a given ORACLE userid who shares one or more tablespaces with other users.

Large sites isolate the impact of hardware failure by separating projects into tablespaces that can be kept on disk drives dedicated to the individual projects. By isolating project A data files onto one set of drives and project B data onto another, failure of one of A's drives does not necessarily wreck the operation of project B. If projects A and B had been distributed evenly across two disk drives, failure of either drive would have guaranteed paralysis of both projects. For small projects, the technique of failure isolation can contradict the load balancing goal, but failure isolation and load balancing can be achieved together at the cost of owning large numbers of disk drives.

Separation of database segments by project minimizes tablespace freespace fragmentation brought on by project expiration. Projects that are funded independently tend to be completed at different times. If one of two or more loosely coupled projects living in a common tablespace is completed, the normal reaction is to attempt to reclaim the tablespace consumed by the expired project. If the freed fragments of freespace are not mostly contiguous, much of the new freespace could be in pieces small enough to be unusable by the extending segments of the ongoing project.

Separation of segments into different tablespaces can help reduce the time taken to recover from media failure. To minimize recovery time, ORACLE administrators commonly take hot backups more frequently for projects with high activity. Because collections of segments can be made available for backup only at the tablespace level, the experienced administrator is thus motivated to devote one or more entire tablespaces to a project, rather than place multiple projects into a single tablespace.

Administer tablespaces

Create only the following **SYS**-owned segments in the **SYSTEM** tablespace:

- database dictionary segments created by **sql.bsq** during execution of the create database statement
- a maintenance rollback segment called **R0**

Administer tablespace resource privileges such that no user except **SYS** has global database resource.

Global database resource provides unlimited privilege to create segments anywhere in the database. The ORACLE database administrator should never grant this type of resource because it permits a user to create any segment in any tablespace, including **SYSTEM**. A user with this privilege can unwittingly incur freespace fragmentation in tablespaces where repair could be expensive. The only user permitted to create segments anywhere in the database, should be **SYS**, the owner of the database dictionary. Consequently, **SYS** is the only user permitted to keep the explicit global database resource.

Note that an ORACLE userid with the **DBA** privilege implicitly holds the **RESOURCE** privilege. Thus a **DBA** user is permitted to create any segment anywhere. This fact is not a particularly dangerous one if the database creator ensures that only the **SYS** user has a default tablespace of **SYSTEM**.

No user except **SYS** should be allowed to use **SYSTEM** as its default tablespace.

Data Dictionary Language (DDL) statements such as those that create and extend tables perform more slowly as the database dictionary table that stores free extent information grows larger. Tablespace freespace fragmentation in the **SYSTEM** tablespace is particularly expensive because it can be repaired only by rebuilding the database. Establishing a non-**SYSTEM** default tablespace for each userid in the database significantly reduces the possibility of irreparable tablespace freespace fragmentation by prohibiting placement of segments in **SYSTEM** that could subsequently be dropped.

A userid without the DBA privilege will be able to create segments only in the tablespaces on which that user has been granted tablespace resource. Since no userid will have resource on the **SYSTEM** tablespace, there is no capacity for the user to place segments there.

A userid with the DBA privilege will be able to create segments in **SYSTEM** only if the user explicitly states "tablespace system" in a segment creation statement.

Ensure that each user in the database uses a tablespace dedicated to temporary segments for its temporary tablespace. Allow no user to use **SYSTEM** as its temporary tablespace.

Use default storage parameters for the temporary tablespace of $\text{next}=\text{initial}=ks+b$, where $k=1,2,3,\dots$; s is the value of **sort_area_size** for the instance; and b is the value of **db_block_size** for the database.

The ORACLE kernel program executes a sort\merge on behalf of any user executing a **select...order by** or other statement requiring row ordering, such as **create index**, **select...distinct**, or **select...union**. The value of the **sort_area_size** instance parameter defines the largest memory area that will be used to perform a user's sort. Any sort that requires more than this much memory is executed in two or more sort runs stored in a temporary segment and merged together to produce the final sorted result.

Temporary segments for a user are automatically created in the tablespace named as that user's temporary tablespace in the **alter user** command. They are automatically dropped upon completion of the sort that motivated their creation.

Sort runs are written to a temporary segment in units of one memory sort area. To maximize the efficiency of the temporary extent allocation process, the DBA should ensure that each temporary segment extent will hold a whole number of memory sort areas.

This is accomplished by setting the default **initial** and **next** storage parameter values to be integral multiples of one sort area size, plus one block, so that both one sort area and one segment header can fit into the temporary segment's initial extent. Selecting temporary tablespace default storage parameters according to this formula minimizes the number of extent allocations and minimizes storage waste.

For example, assume that the **sort_area_size** for the *sid1* instance is 256 Kb, and that a given database user is executing a sort that requires 1,985 Kb of sort area. If the standard default storage parameters of **initial** 10k, **next** 10k, and **pctincrease** 50 were in force for the **TEMP** tablespace, the extent allocations for the temporary segment required to execute the sort in this 2 Kb-block database would be those shown in the next table.

Note that with the standard default storage parameters, this sort would require twelve extent allocations. Also note that this segment consumes 609 Kb more than would have been required to perform the sort. Waste is calculated as the amount actually consumed minus the amount actually required, which is the storage required for the sort plus one 2 Kb segment header block: $2,596 - (1,985 + 2) = 609$ kilobytes.

Extent	Extent Size in 2 Kb Blocks	Segment Size in 2 Kb Blocks	Segment Size in Bytes
0	5	5	10Kb
1	5	10	20
2	8	18	36
3	12	30	60
4	17	47	94
5	26	73	146
6	38	111	222
7	57	168	336
8	86	254	508
9	129	383	766
10	193	576	1152
11	289	865	1730
12	433	1298	2596

By contrast, the following table shows that with temporary tablespace default storage parameters set to **initial 514k next 514k pctincrease 0** (using $ks+b$, where $k=2$, $s=262,144$, and $b=2,048$), the same sort would require only four extent allocations, with a total waste of only 69 Kb.

Extent	Extent Size in 2 Kb Blocks	Segment Size in 2 Kb Blocks	Segment Size in Bytes
0	257	257	514Kb
1	257	514	1028
2	257	771	1542
3	257	1028	2056

Temporary segments are created and dropped rapidly during the normal operation of a large and busy ORACLE system. Any time a database segment is dropped, it incurs at least one unit of tablespace freespace fragmentation. In alignment with the constant pursuit of zero freespace fragmentation, the professional DBA should never allow temporary segments to be created in any tablespace other than one specifically

Create rollback segments

designed to hold them. Under no circumstances should the temporary segments be allowed to pollute the **SYSTEM** tablespace. The DBA can query the **dba_users** view to ensure that each user's temporary tablespace is set to **TEMP** or **TEMP_n**.

A rollback segment holds the contents of a database segment's original value while a change awaits a commit. Rollback segments record the information necessary to allow a user to roll back a transaction, meaning long-running queries can use this time-stamped data to reconstruct how a table looked at the moment in time when that query started, even if the actual contents of that table have subsequently changed. Thus, rollback segments give the abilities to roll back transactions and to produce read-consistent queries.

Note: The DBA may not create rollback segments in a non-**SYSTEM** tablespace without first creating a companion rollback segment to the rollback segment named **SYSTEM** in the **SYSTEM** tablespace.

For each ORACLE instance, create a private rollback segment named **R0**, owned by **SYS**. List **R0** in the **rollback_segments** instance parameter value whenever rollback segment maintenance is taking place, and omit **R0** from this list whenever production rollback segments are online. Do not drop **R0**.

Rollback segment names serve the sole purpose of distinguishing one rollback segment from another. Since rollback segments are always easily identifiable by context, it is not necessary to give rollback segments connotative names. The name **R0** was selected for its economy.

The additional **SYSTEM** tablespace rollback segment required to add tablespaces is also named **R0** and uses the following storage parameters:

- **initial** 10K
- **next** 10K
- **pctincrease** 0
- **minextents** 2

Dropping **R0** will fragment the **SYSTEM** tablespace freespace and require you to re-create the maintenance rollback segment on your next round of rollback segment maintenance. However, it is important to take **R0** offline for normal database operation by omitting its name from the **rollback_segments** instance parameter value. Failure to take **R0** offline for production use will lead to consumption of **SYSTEM** tablespace freespace by user transactions, which will ultimately lead to expensive fragmentation of the tablespace.

Production rollback segment names match the pattern *Rnn*, where *nn* is a unique two-digit number in the range from 01 to 50. There is one rollback segment for approximately every four concurrently active transactions in a mixed batch OLTP environment. Each production rollback segment uses storage parameters:

- **initial** 20K
- **next** 20K
- **pctincrease** 0
- **minextents** 2

Transaction requirements dictate the sizes of the rollback segments. There are multiple homogeneously-sized extents allocated for each rollback segment.

For applications that do not allow a choice of which rollback segment to use for large transactions, make all rollback segments the same size.

Options for Customizing your Installation

Software/Database Maintenance Option

After you have completed a successful installation you can continue using Installer to perform various maintenance activities on your products. The Software/Database Maintenance option on the Install Actions Menu provides you with the following options:

Product Administration This option allows you to perform administrative tasks such as redefining your terminal definition and regenerating forms and resource files.

Relink Product Executables This option allows you to relink your product executables if, for instance, you have added a new network driver to a product or upgraded your operating system and system libraries. See "Relinking" later in this chapter.

Create/Upgrade Product Database Objects This option allows you to reinstall tables for your products.

Note: When using the Software/Database Maintenance option, product dependencies are not activated.

Software Deinstallation Option

This option allows you to deinstall ORACLE software that has been installed with the Installer only.

Environment Variables for Installation

Experienced users may want to customize the installation by setting some of the environment variables that are described below. Sample values for each variable are given in the first line of each section.

DBS_SIZE

DBS_SIZE=10M

You can set this variable to specify the size of a new database to be created. You must use a format acceptable to the **create database** command (for example, 2000K, 2M, and so on).

Make sure that your database file is large enough to hold all your user data, plus a margin for overhead and indexes. (Also set the **LOG_SIZE** environment variable, to adjust the size of the log files.)

ORACLE_BASE

ORACLE_BASE=*/home/dba/oracle

The **ORACLE_BASE** environment variable defines a base directory for the *oracle* user. For more information, refer to "ORACLE DBA Recommendations" in the chapter "Preparing to Install Oracle Products."

ORACLE_HOME**ORACLE_HOME=/usr/oracle**

The **ORACLE_HOME** environment variable defines the home directory for the *oracle* account. You will also have the opportunity to set this value interactively when you run the Installer.

ORACLE_PATH**ORACLE_PATH=/usr/oracle/bin**

This **ORACLE** environment variable tells **SQL*Plus**, **SQL*Forms V3.0**, and **SQL*Menu V5.0** where to find unspecified files, including product files. In particular:

- **For reading a file:** If you specify a file to be read, and do not specify an absolute directory in the filename, the application will search for a file of that name first in a directory specified by **ORACLE_PATH**, then in the current directory. This includes invoking the product without specifying a directory.

SQL*Menu V5.0 will search the current directory, then **\$ORACLE_HOME/menu5/libs**, then the path specified by **ORACLE_PATH**.

- **For writing a file:** If you are creating or writing to a file, and do not specify an absolute directory in the filename, the application will search for a file of that name in a directory specified by **ORACLE_PATH**. If the file is there it will write to it. Otherwise, it will write to the current directory (creating a new file if one does not already exist there).
- **If **ORACLE_PATH** is not set:** The application will always read from and write to the current directory.

ORACLE_SID**ORACLE_SID=sid**

The **ORACLE_SID** environment variable defines the system identifier (the *sid*). You will also have the opportunity to set this value interactively when you run the Installer.

NLS_LANG**NLS_LANG=American.US7ASCII**

This optional environment variable defines the language and character set in which you want to run the Installer.

NO_MAKE**NO_MAKE=TRUE**

This optional environment variable should be set to **TRUE** if your system is without such development executables as **make**, **cc**, and **ld**.

Customizing the init.ora File

The **init.ora** file is a database configuration file that is read every time ORACLE is started up. This file contains a number of parameters that determine various characteristics of the system. For a complete discussion of this important file, refer to the *ORACLE7 Server Administrator's Guide*.



The *ORACLE7 Server for UNIX Administrator's Reference Guide* documents the default **init.ora** file that comes with your SunOS system. This file includes parameter values for small, medium sized, and large databases. The values that do not apply to you can be commented out.

Note that the complete name of the file that is read on database startup is **initsid.ora**, where *sid* is the system identifier of the database in question. In this manual the file is referred to simply as **init.ora**.

If you wish to customize the **init.ora** file when you install ORACLE7 Server (in order to customize the database), choose the "Install Software Only" option to install the software first. Then customize the **init.ora** file, and only then use the "Create New Database Objects" option to create the new database. Note that if you simply choose "COMPLETE SOFTWARE/DATABASE FRESH INSTALLATION" to install the software from tape, you will not have the opportunity to modify the **init.ora** file before the database is created.

Relinking

The Installer offers you the option to relink the executables. This section explains why you might choose to do so, and provides detailed information about relinking.

When to Relink

The Installer asks whether you would like to use the default ORACLE executables or to relink on site. The default is not to relink.

Under any of the following conditions, however, you *must* relink:

- if you are installing non-default network drivers (the defaults are the pipe, fast, and TCP/IP drivers)
- when there is multiple TCP/IP vendor support, as with your SunOS system
- if you are installing Multi-Client Server
- when the user's *dba* group is not *dba*

Install Space Considerations

If you choose to relink executables during your install (to include SQL*Net drivers in your executables), you will need more space on the file system where your ORACLE home directory resides.

Also, relinking will require approximately 5500 free blocks of temporary space in the */tmp* file system since linking creates intermediate files in */tmp*.

It is possible to designate a different temporary directory by setting the **TMPDIR** environment variable to a directory that is writeable by the *oracle* user and has enough free space. For instance,

```
TMPDIR=/usr/tmp; export TMPDIR
```

will cause the C software generation programs to use */usr/tmp* for creation of temporary files rather than */tmp*.

Note: Under the C shell, use the command **setenv TMPDIR /usr/tmp**.

Also, a relink of ORACLE executables will rename some of the old executables and save them rather than removing them. Once the new executables are in place, there is no need for the old ones. They are found in the **\$ORACLE_HOME/bin** directory with "O" appended to their original names (i.e., **exp** is renamed **expO**). Remove them if you are short of disk space.

Installing ORACLE in a Network Configuration

To set up ORACLE in a network configuration, install ORACLE on the server and not on client machines.

Begin by deciding just where on the server to install ORACLE. Normally you will install ORACLE on an existing partition of the server's disk that is mounted on the diskless workstations.

The directory with all the binaries (containing `$ORACLE_HOME/bin` and `$ORACLE_HOME/dbs`) is referred to when you run the utilities. For this reason the executables must be installed in a partition that is available to clients over NFS. A partition is available to a diskless workstation only if it is "mounted" on the diskless workstation.

Mount a partition by listing it in `/etc/fstab`. A sample `fstab` entry looks like this:

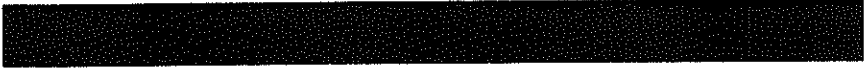
```
honcho3:/usr.MC68020 /usr nfs rw 0 0
```

Example

Here is an example of a client/server installation:

1. If this has not already been done, mount the server's `/usr` partition on the client as `/usr` over NFS.
2. In `/usr` on the server, create an oracle subdirectory.
3. Install ORACLE in `/usr/oracle` on the server, using the automated installation procedure given the "Oracle Installer" chapter.
4. Now set the client's `ORACLE_HOME` variable to `/usr/oracle`, set the `ORACLE_SID` appropriately, and set `PATH` to include `/usr/oracle`. Set the `TWO_TASK` variable to T: for TCP/IP.
5. To verify that the configuration is set up correctly, try to run SQL*Plus from the client machine.

It will find the `sqlplus` executable in `/usr/oracle/bin` on the server, via NFS. Then it will connect to the database on the server using SQL*Net TCP/IP.



PART 2: ADMINISTRATION

6

ORACLE SYSTEM ADMINISTRATION ON SunOS

This chapter describes SunOS-specific aspects of ORACLE7 Server system administration and tuning. It includes the following topics:

- shared memory parameter adjustments
- relocating the SGA
- running ORACLE in a networked configuration
- performance tuning
- configuring disk I/O for better performance
- asynchronous I/O and multiple DB writers
- raw disk devices
- buffer cache size



This information complements other, more general, discussions of tuning and system administration. For a complete discussion of system administration and tuning, refer to the *ORACLE7 Server Administrator's Guide*. For additional information on tuning UNIX-based systems, see the *ORACLE7 Server for UNIX Administrator's Reference Guide (SARG)*.

Shared Memory Parameter Adjustments

If shared memory is not sufficient, you will not be able to install the ORACLE Server.

You must ensure that your machine's shared memory configuration is adequate to handle the SGA structure that will be generated by your ORACLE system. The default value may not be large enough. Refer to the section titled "Shared Memory Parameters" in the "Key to ORACLE on SunOS" chapter for more information.

In particular, you must check the kernel parameters corresponding to:

- the maximum size of a shared memory segment
- the maximum number of shared memory segments in the system
- the maximum number of shared memory segments that a user process may attach
- the maximum amount of shared memory that may be allocated system-wide

tstshm

You can use the **tstshm** executable to evaluate the existing shared memory configuration. A detailed discussion of shared memory parameters and their recommended values appears in the "Key to ORACLE on SunOS" chapter. See the "Configuring your ORACLE System" chapter for information about configuring your system.

Relocating the SGA

The System Global Area (SGA) is the ORACLE structure that resides in shared memory and contains, among other things, static data structures, locks, and data buffers. The address at which the SGA is attached affects the amount of virtual address space available not only for database buffers in the SGA, but also for such things as cursors in the user's application data area.

Do not attempt to rebuild the oracle executable without first consulting Oracle support personnel.

Procedure

After consultation, use the following steps to relocate the SGA.

1. **Determine the valid virtual address range for attaching shared memory segments.**

This can be done by invoking the shared memory test program, `tstshm`, with no arguments.

In the resulting `tstshm` display, two lines indicate the valid range ("Lowest shared memory address" and "Highest shared memory address"), as shown in the following example. (Different systems will show different values from the one in this example.)

```
Lowest shared memory address  = 0xbc00
Highest shared memory address = 0x156bc00
```

2. **Determine the valid virtual address boundaries at which a shared memory segment can be attached.**

For example, some systems require that shared memory segments be on 512-byte boundaries. This information can be obtained from the "Segment boundaries" output of `tstshm`.

3. **Determine the size of your SGA.**

The size is displayed next to the heading *Total System Global Area* when your database system starts.

4. **Move to the \$ORACLE_HOME/rdbms/lib directory, and run `genksms` to generate the file `ksms.s`.**

```
$ genksms > ksms.s
```

5. **Raise or lower the value of the symbol `sgabeg` by editing the file `ksms.s`.**

For example, the `ksms.s` file looks like this:

```
sgabeg = 0xe0000000
.global _ksmsgf_
_ksmsgf_ = sgabeg+0
.global _ksmvsg_
_ksmvsg_ = sgabeg+4
.
.
.
```

Here, the `sgabeg` symbol specifies the address at which the SGA will be attached. Change the line

```
sgabeg = 0xe0000000
```

to reflect the new address at which you want to attach the SGA:

```
sgabeg = address
```

The `ksms.s` file contains a warning message about editing. This message can be ignored when you are relocating the SGA.

6. **Shut down the existing ORACLE database.**
7. **Rebuild the oracle executable by entering:**
8. **Save the old oracle executable if you wish. Then move the new oracle executable to `$ORACLE_HOME/bin` and change its permissions as follows:**

```
$ make -f oracle.mk oracle
```

```
$ cd $ORACLE_HOME/bin
$ mv oracle oracle.old
$ cd $ORACLE_HOME/rdbms/lib
$ mv oracle $ORACLE_HOME/bin
$ chmod 4751 $ORACLE_HOME/bin/oracle
```

The result is a new ORACLE kernel that loads the SGA at the address specified by `sgabeg`.

Running ORACLE in a Networked Configuration

Do not mount your ORACLE database files remotely over a network file server (NFS). NFS does not support synchronous write capabilities, and thus does not provide facilities to guarantee data integrity over a file network.

NFS is not appropriate because it involves a database spanning remote disks. When you perform an update across a network you have no confirmation that the remote update actually occurred. In addition, such a setup is extremely overhead-intensive. When the detached processes run on a machine other than the one where the files are located, the processes must constantly read and write large amounts of data across the network to access each database block needed and to write to log files. This places a very heavy load on the network. Contrast this to a system in which the detached processes run on the system where the files are located, and the user logs in over SQL*Net. Here, only the SQL statements and data returned from them must pass over the network. By comparison, this is a very small amount of network overhead.

An additional problem can arise when the detached processes run on a machine other than the one where the files are located. Users on the system containing the files may not realize that the database is up, since no processes are running on their machine. They could attempt to bring up the database on their machine, causing problems with each instance that is up as well as possible database corruption.

To run ORACLE in a networked configuration, you can access executables via a network file server (NFS). However, you must access the remote database using SQL*Net because the default two-task driver uses pipes, and pipes cannot span CPU boundaries.

Process Structure

In a networked configuration using TCP/IP, the ORACLE executables exist on a server machine where they are accessed by client machines. The figure below shows ORACLE process architecture under a standard UNIX file server installation reflecting the system's diskless architecture. The application itself resides in the server's file system; the application process resides on the diskless system. The "Installation Tutorial" chapter provides instructions for installing ORACLE in a network configuration.

Note: The server in this instance might be a Sun SPARC station 10; the diskless system might be a Sun 4/50 workstation.



Careful tuning can dramatically increase the speed of the ORACLE7 Server on your SunOS system. This section explains how to monitor and tune your system for optimal performance of ORACLE. For a general discussion of performance and tuning issues, see the *ORACLE7 Server Administrator's Guide*.

This section covers the following topics:

- SunOS-specific tunable parameters
- monitoring performance

SunOS-specific Tunable Parameters

This section covers SunOS-specific parameters that are important in tuning ORACLE on your system. Some of these are described in the *ORACLE7 Server Administrator's Guide*. The others are new and therefore have not been described before.

I/O parameters

db_block_compute_checksums

The *init.ora* parameter **db_block_compute_checksums** controls whether a checksum is put into every block before the block is written to disk. The default is **false**. If you set this parameter to **true**, there is more overhead. (A block read validates an existing checksum whether or not this option is enabled.) A block is marked as corrupt if a checksum fails.

db_writers

This parameter specifies the number of database writers to be brought up when the database is started. The default value is 1.

The setting of this parameter depends on whether you use SunOS asynchronous I/O (AIO) or Oracle multiple database writers. Please consult the "Asynchronous I/O and Multiple DB Writers" section later in this chapter to decide which option to use.

If you decide to use multiple database writers, you should set this parameter to the number of database writers you desire and either unset the **async_write** parameter or set **async_write=FALSE**. The number of database writers needed is application specific. In general, however, this number should be at least equal to the number of disk drives that are used for holding tablespaces. Note that each additional **dbwr** process will consume one more semaphore on the system, so the **processes** parameter in the *init.ora* file and the SunOS semaphore parameters may

need to be raised accordingly. You will also want to set **db_block_write_batch** to a value twice that of **db_writers**.

async_write

This parameter determines whether ORACLE uses SunOS asynchronous I/O (AIO). Please consult the "Asynchronous I/O and Multiple DB Writers" section later in this chapter to decide whether to use SunOS AIO.

If this parameter is not set (default) or set to **FALSE**, the SunOS AIO feature is not enabled for ORACLE. To turn on the SunOS AIO feature for ORACLE, set **async_write=TRUE** and **db_writers=1** in the **init.ora** file.

db_block_write_batch

This **init.ora** parameter, described in the *ORACLE7 Server Administrator's Guide*, sets the number of blocks scheduled simultaneously for writing.

The value of this parameter should not be greater than the number of asynchronous I/O buffers. The default is 8.

As of ORACLE V6.0.27, this parameter is the basis for values previously set by the following **now-obsolete** parameters:

- **db_block_max_mod_pct**
- **db_block_max_clean_pct**
- **db_block_timeout_write_pct**
- **db_block_max_scan_pct**

db_file_multiblock_read_count

This parameter is used for multi-block read. It sets the number of blocks to read when doing I/O during a sequential scan. It is a useful parameter for performing a full table scan in which the **WHERE** clause does not refer to an indexed column. The range is 0 – 32; the default is 8.

use_readv

ORACLE supports multi-block read. The **use_readv** parameter enables you to choose whether or not to use the **readv** system call. If you set **use_readv** to **true**, then multi-block read is done through the **readv** call to perform a scattered read. If you set **use_readv** to **false**, then a buffer will be allocated and scattered read will be simulated by reading all blocks, followed by memory-to-memory copies.

Latches protect shared data structures in the SGA. A process acquires a latch when working with a structure, and drops the latch when finished with the structure. Each latch protects a different set of data, identified by the name of the latch.

Note that the **TIMEOUTS** column in the **SQL*DBA** display is system-specific in its meaning. The number of **TIMEOUTS** is the number of times the process was unable to get a resource because it could not get a latch for that resource. (This number will be different on different machines.)

Note: The **latch_wait_time**, **latch_wait_mult** and **max_backoff** parameters have been obsoleted as of Release 7.0.12.

cpu_count

This parameter specifies the number of CPUs available to ORACLE on a multiprocessor (MP) machine.

On Sun machines, the value of **cpu_count** is set internally and automatically by ORACLE. Do **not** specify a value for this parameter in the **init.ora** file.

log_simultaneous_copies

This parameter specifies the number of copy latches. If this is not specified, the default value is set to **cpu_count** (described above) or set internally. Note that the **SQL*DBA show parameters** command will show the value of this parameter as the default (which is 0), rather than the internally determined value.

spin_count (for machines with multiple CPUs only)

This parameter sets the number of times an ORACLE process attempts to get a latch before sleeping. If the ORACLE process tries to get a latch and finds that it is busy, ORACLE spins and checks back with the latch up to **spin_count** number of times until the latch is not busy. Once it attains **spin_count**, the ORACLE process sleeps for a set period of time, then wakes up and tries again.

You can adjust performance by changing the value of **spin_count**. If you use a high value, you will get the latch sooner than if you use a low value. However, you may use more CPU time spinning to get the latch if you use a high value for **spin_count**.

With regard to tuning, note that the default value is 0 (that is, no spinning), which should be the setting for single-processor machines. A reasonable **spin_count** value is 2000.

If there is a lot of latch contention and it is unlikely that a latch will be acquired during the spin, set this number low so that the process can relinquish the CPU to other processes.

ORACLE block size

db_block_size

This **init.ora** parameter sets the size in bytes of ORACLE database blocks. See the *ORACLE7 Server Administrator's Guide* for a discussion of choosing block size. The "Key to ORACLE on SunOS" chapter describes the block size range and default value on your system.

Redo log files

The following parameters set values relating to log archiving that affect performance.

log_allocation

This **init.ora** parameter indicates the number of redo log file blocks allocated to an instance each time it requires more space in a current online log file. (See **log_buffer** for a discussion of how redo log buffers are organized.) The default is 200 blocks. Set this at least equal to the size of the online redo log file.

log_blocks_during_backup

This parameter controls whether or not block images are written to the redo log during a hot backup. The default is **true** for SunOS systems. This may be set to **false** if the ORACLE block size equals the O/S physical sector size, or if it is otherwise ensured that hot backup reads consistent versions of blocks even if those blocks are being written at the time. (Put another way, this may be set to **false** on machines that can guarantee the atomicity of a single block I/O request during a hot backup.)

log_buffer

This parameter sets the number of bytes allocated to the redo log buffers in the SGA. In general, larger values reduce redo log file I/O, particularly if transactions are long or numerous. The default is four times **db_block_size**.

Writes to the redo log buffers include the block of altered bytes, the rollback segments block, and the database header data. These are therefore generally larger than database buffers.

log_checkpoint_interval

This **init.ora** parameter sets the number of redo log file blocks (O/S blocks, not ORACLE blocks) required to trigger a checkpoint. A checkpoint ensures that all blocks modified since the previous checkpoint are actually written to disk. A checkpoint occurs when either the redo log file is full or the **log_checkpoint_interval** is reached. The default value is 320.

Refer to the *ORACLE7 Server Administrator's Guide* for further discussion of performance and tuning issues related to setting checkpoints.

log_files

This parameter sets the maximum number of redo log files that can be opened at runtime for this database. This is in effect every time an instance opens a database, and lasts until the database is closed.

This parameter is related to the **maxlogfiles** argument for the **create database** statement, which is the *absolute* maximum number of redo log files for all instances. If neither **log_files** nor **maxlogfiles** is used, the default limit to the number of redo log files is the ORACLE limit of 255 files. Otherwise, the default is 16 files.

Monitoring Performance

You can use certain tools, as well as the **SQL*DBA monitor** command, to observe the behavior of the database.

To evaluate performance you need to look at:

- I/O activity
- CPU usage
- paging

I/O Activity. Disk drives and controllers have a maximum I/O throughput, both in terms of I/O per second, and blocks per second. When evaluating your system performance, make sure you have not reached the limits of the I/O bandwidth.

CPU Usage. Determine whether your system is CPU-bound. If there is no idle time, then you have reached the limits of your CPU on your computer.

Paging. For maximum performance, there should be very little paging in the normal course of processing. Refer to your system administrator's manual for a discussion of paging.



For information on monitoring I/O activity, CPU usage, and paging, see the "Monitoring and Tuning" chapter of the *SARG*.

Using SunOS tools

Several tools are available to monitor the performance of ORACLE for the SunOS operating system.

- **ps**
- **iostat**
- **perfmeter**

These tools give statistics for CPU usage, interrupts, swapping, paging, and context switching, for the entire system, not just the ORACLE processes. For more information about these commands, refer to your system administrator's manual.

Configuring Disk I/O for Better Performance

The following sections describe the options available on SunOS for configuring disk I/O for better performance. These topics include:

- asynchronous I/O and multiple DB writers
- raw devices
- buffer cache size

Asynchronous I/O and Multiple DB Writers

Asynchronous I/O (AIO) is a standard SunOS feature. SunOS AIO allows ORACLE to write to multiple disks simultaneously, thus increasing disk throughput. With SunOS AIO, multiple database writes are sent to multiple disk drives without waiting for the previous writes to finish. For example, with 10 disk drives, 10 asynchronous writes can take place in the same amount of time as a single synchronous write.

Since Oracle7 for SunOS supports SunOS AIO, having multiple database writers is usually not necessary. In other words, when SunOS AIO is used, Oracle7 for SunOS can write to many disks simultaneously using only one database writer process.

However, SunOS AIO only works on raw devices. Therefore, if any of your database files exist on file system partitions, you may want to use multiple database writers instead. Having multiple database writer processes is a software simulation of AIO. To use multiple database writer processes is a software simulation of AIO.

Oracle7 for SunOS does not use SunOS AIO by default. To use SunOS AIO, set the parameter `async_write=TRUE` in the `init.ora` file. Note that the `db_writers` parameter must be set to 1 when SunOS AIO is used.

To use multiple database writers, set the `db_writers` parameter in the `init.ora` file. Note that the `async_write` parameter should be either unset or set to `FALSE` when multiple database writers are used.

Raw Devices

A raw device, or raw disk partition, is a hardware device that is supported by a character device driver. A character device driver accesses the raw device through the special files that are in the `/dev` directory (see `mknod` and other information about raw devices in the operating system documentation).

These devices are not buffered by the SunOS kernel; data is transferred directly between the user's buffers and the device. Disk partitions are among the SunOS devices that are supported by character device drivers. All references in this manual to "raw devices" refer specifically to disk partitions supported by character device drivers.

It is commonly suggested that bypassing UNIX file buffering by using raw devices can result in a five to twenty percent improvement in ORACLE disk I/O. These figures do accurately reflect the performance of ORACLE disk reading and writing. However, it is *not* correct to infer a similar gain in general performance or throughput anywhere but in the most transaction-intensive database.

Advantages and Disadvantages

The performance advantages of using "raw" (character special) files for the ORACLE database (or redo log) files are:

- I/O is done directly from the disk to the SGA, with little CPU overhead. (ORACLE bypasses the SunOS buffer cache.)
- The overhead of SunOS read ahead is avoided.
- The overhead of the SunOS file system is avoided.
- The SunOS and ORACLE buffer caches are independently tuned.
- You may reduce the size of the SunOS buffer cache and free memory for other uses.

Bypassing UNIX file buffering results in a savings on every disk read or write. This savings shows up as a throughput improvement only if disk I/O is the system performance bottleneck.

The memory used by UNIX to buffer file I/O can be better used by the database that does its own I/O and caching. The more memory a machine has, the less effective an argument for going raw this memory savings becomes.

Certain problems are involved in the use of raw devices. For example:

- Raw devices may be difficult to set up.
- Backup procedures are more complex than for the standard file system.
- Optimal performance requires tuning both the SunOS buffer cache and the ORACLE buffer cache. You may be able to use raw devices for database access. (If the **limit** parameter exists, it will be ignored.) This approach is attractive due to performance gains that result from circumventing normal SunOS file system I/O.
- Your SunOS system administrator must be involved in all database installations and reconfiguration.
- When raw disk devices and SunOS files are mixed within an ORACLE database, the SunOS files must still be within the value of the **limit** parameter.
- This does not solve problems with **limit** that may arise when exporting tables larger than two Gigabytes. Such tables would also have to be exported to raw devices (such as another disk partition).
- This does not solve problems with **limit** that may arise when reading the contents of the ORACLE distribution media onto the disk.
- The operating system will not protect cylinder 0 if it is part of a raw partition. To prevent this, make sure the raw partition used by ORACLE **does not** start at cylinder 0.

Clients with small databases usually do not have the luxury of choosing from a sufficient number of well-sized raw device sections. Disk sections usually come in odd sizes that do not lend themselves to the implementation of a good database architecture.

Upon finding that a particular disk drive is "hot" and that performance would benefit from movement of an ORACLE data file from that drive to some other, it is likely that no acceptably sized section exists on a "cool" drive. Moving data files around, a simple and attractive option in a UNIX file system environment, is potentially impossible with raw devices.

Adding space to a tablespace can be a difficult process in a raw device environment. Normally, all raw sections are assigned data files at initial configuration time, leaving no raw storage available to accommodate normal tablespace growth.

Criteria for Using Raw Devices

An ORACLE architect/administrator should choose to use raw devices only if each of the following criteria hold:

- transaction volume is high
- raw disk sections are plentiful
- disk volume is large

Transaction volume

Use raw devices for ORACLE files only if the site has sufficiently high transaction and query volume that disk I/O is the performance bottleneck.

If disk I/O is the performance bottleneck, then it is likely that the highest throughput gain lies in the SQL trace performance analysis of a few individual application SQL statements. Normal use of queries that return a large number of rows (tens or hundreds of thousands) have motivated many DBAs to use raw devices. However, before jumping to raw devices, the DBA should determine if an application that processes tens or hundreds of thousands of rows is designed as well as it should be.

Sufficient sections

Use raw devices for ORACLE files only if the site has at least as many raw disk sections (partitions) as it will have ORACLE tablespaces.

Without at least as many raw sections as tablespaces, the DBA is forced to integrate segments with incompatible fragmentation characteristics. This hurts performance more than the use of raw I/O helps it. All ORACLE databases should contain at least these tablespaces:

SYSTEM	SYS owned dictionary segments only
RBS	rollback segments only
TEMP	temporary segments only
TOOLS	SYSTEM owned segments only (plus, crt, forms, srw, etc.)
USERS	users' personal tables, etc.

Naturally, many databases will have multiple applications housed in multiple tablespaces, and some DBAs will separate applications data and indexes into different tablespaces. The greater the number of tablespaces in the database, the greater will be the number of disk sections required to effectively use a raw device architecture.

Disk volume

Use raw devices for ORACLE files only if the site has enough disk space that it can afford over-allocation of small ORACLE tablespaces.

For those sites that do go raw, fully flexible disk load balancing is possible only if disk section used for raw ORACLE storage are all the same size. Living by this homogeneous sizing requirement costs disk space because it forces over-allocation of small tablespaces.

For example, if a site's data dictionary requires only 30 Mb, yet the site uses 150 Mb raw sections, then there will be 120 Mb of wasted space in the **SYSTEM** tablespace. Using the extra space in **SYSTEM** for non-dictionary segments can seem preferable to explaining why 120 Mb of disk space sits idle, especially if the DBA has to convince a finance committee that the site needs to buy more disk drives.

Administrators using raw devices will also experience more difficulty in adding space to existing ORACLE tablespaces. To continue our example, it would be difficult to resist the temptation to pre-allocate as many as 150 Mb chunks of disk space as possible to ORACLE tablespaces. But, if after a month or two of operation the administrator finds the need to add a data file to a tablespace, the options become buy a new disk drive, re-create the database with a new, more appropriate tablespace architecture, or add a data file from space available in the UNIX file system.

The first two options are immediately and obviously expensive. The last option initiates the DBA into the business of conducting two synchronized operations for each cold database backup.

Using Logical Volumes

The concept of logical volumes has been introduced on several UNIX platforms. Logical volumes, or logical disks, are a level of abstraction above raw devices. A logical disk can consist of multiple raw device partitions from on or more drives. In other words, the size of a logical volume is not restricted to one physical drive, but can span several physical drives.

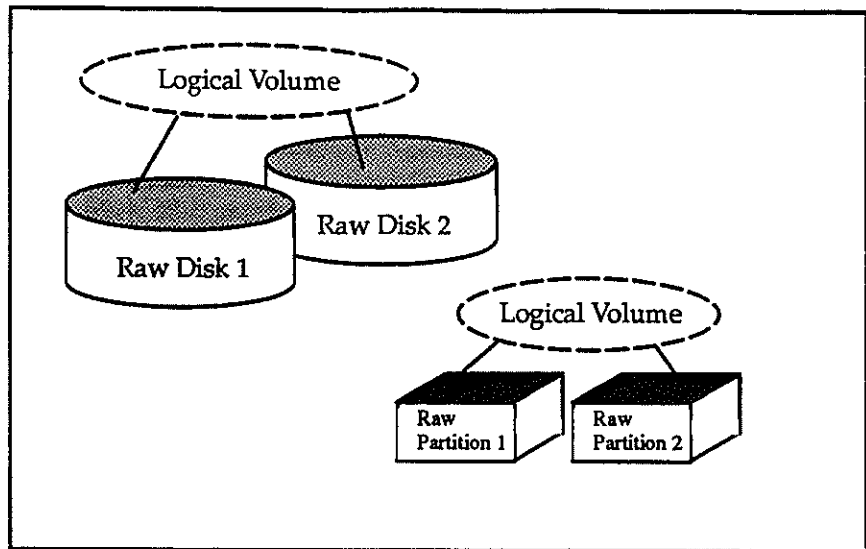


FIGURE 6-1 Logical volumes

Logical volumes can alleviate most of the disadvantages associated with raw devices, and offer the following benefits:

- easier configuration planning
- dynamic performance tuning
- mirroring and online disk replacement

Configuration planning

With logical volumes, you can create logical disks based on raw partition availability, since logical disks can span multiple disk drives. The disk drives do not have to be re-formatted in order to obtain appropriate logical disk sizes.

Dynamic performance tuning

Disk performance can be optimized when the database is online by moving "hot spots" to cooler drives. Most hardware vendors who provide the logical disk facility also provide a graphical user interface that can be used to perform this tuning.

Mirroring and online disk replacement

Mirroring of logical volumes is possible and should be used to protect against loss of data. If one copy of a mirror fails, dynamic re-synchronization is possible. Some vendors also provide the ability to replace drives online in conjunction with the mirroring facility.

Setting Up Raw Devices

You should not attempt to set up a raw disk device without an experienced system administrator and specific knowledge about the machine you are using.

With these prerequisites, you can use the following general procedure:

1. **Locate the names of the disk devices on the system.**

These are in the `/dev` directory.

The system administrator's guide contains documentation on device names.

2. **Determine the names of the free disk partitions.**

A free partition is one that is not used for a SunOS file system. That is, when you execute the `/etc/mount` command, it is not listed. Also, it is not in use as a swap device and is not in use by other SunOS applications (for example, other instances of ORACLE). Furthermore, a free partition *must not* overlap or be overlapped by a SunOS file system. It must not overlap a swap partition, either.

To find out whether a partition is free, obtain a complete map of the starting locations and sizes of the partitions on the device and check for free space. Note that some partitions may contain file systems that are currently not mounted and are not listed in the `/etc/mount` output.

Note: Make sure that the partition does **not** start at Cylinder 0.

3. **Set up the raw device for use by ORACLE.**

Begin by verifying that the partition is formatted. If it is not, then format it.

Next, make sure that the partition is owned by *oracle*. If necessary, use `chown` to change its ownership. For example:

```
$ chown oracle /dev/rxylg
```

Make the partition accessible only by *oracle*. To do this, use `chmod`. For example:

```
$ chmod 600 /dev/rxylg
```

4. **Create or add the new partition from a new database.**

From SQL*DBA, use the SQL statement `create database` to create the database using the specified raw partition.

Note: The size of an ORACLE database created in a raw partition must be at least two ORACLE block sizes smaller than the size of the raw partition.

For example:

```
create database sid
logfile '?/dbs/log1.f' size 100K, '?/dbs/log2.f' size 100K
datafile '/dev/rxy1g' size 10000K reuse;
```

Alternatively, if you wish to add the partition to a tablespace in an existing ORACLE database, enter:

```
alter tablespace tablespace_name add datafile
'/dev/rxy2g' size 10000K reuse;
```

You can use the same procedure to set up a raw device for the redo log files.

You can have any mixture of file systems and raw disk partitions within a single tablespace.

Using Raw Devices

When using raw devices you must carefully consider optimization of head movement and controller activity. As much as possible, you should balance the load across separate drives and controllers. In a typical environment you must consider four kinds of I/O:

- the SunOS file system
- SunOS swapping and paging
- ORACLE tablespace
- ORACLE log file

When you attempt to balance I/O, follow this procedure:

1. **Separate the redo logs and the database file(s) onto different drives.**

In high transaction rate environments, a great deal of I/O activity will be directed to the redo logs. Ideally, the redo logs will exist alone on a disk drive.

2. Spread heavily used database files over different drives.

If you create several tablespaces composed of files that reside on different drives, you keep "hot spots" from occurring.

Alternatively, you can create a single tablespace composed of several files residing on different disks.

3. Separate tables and their indexes.

If a SQL statement requires access to a table and its index, you can have greater throughput if you put the table and its index on separate drives.

4. Create more than one rollback segment and distribute the segments evenly across the drives.

A high transaction rate environment requires heavier use of rollback segments. If you create additional segments across drives, you not only balance potentially heavy I/O, but you also avoid contention for the rollback segments themselves.

Cache your rollback segments to avoid physical I/O.

5. Separate ORACLE and SunOS I/O.

Ideally, you would put SunOS on a drive separate from ORACLE I/O.

If SunOS and ORACLE must reside on the same drive, put the ORACLE file system files and SunOS file system files as close together on the disk as possible.

You may also put raw partition files on the same drive as SunOS file system files.

Backing Up Raw Devices

Raw device backup can be handled in either of two ways:

- Copy the raw device to a SunOS file and allow the normal backup procedures to copy it.
- Copy the raw disk device directly to tape.

Copying to a SunOS file

If you copy the raw device to a SunOS file and allow the normal backup procedures to copy it, you must still use `dd` to copy from the raw device.

Caution is required because some SunOS backup programs ignore large files; you must verify that the SunOS file is indeed backed up.

Copying to a raw tape device

One way to copy the raw disk device directly to the raw tape device is in the following example:

```
$ dd if=/dev/rxylg of=/dev/rst0
```

This command consults in one block per tape record. It assumes that the database uses the entire disk partition.

The following command shows more typical usage:

```
$ dd if=/dev/rxylg of=/dev/rst0 bs=10b count=10000
```

This command copies 10,000 blocks of the tape, 10 blocks per tape record (assuming 512K database blocks).

Use `dbfsize /dev/rxy1g` to determine the number of ORACLE blocks in the database file. Add one ORACLE block to that number for the database header block.

The `dd` command will typically run faster with a larger block size (`bs=20K` or `bs=32K`). Block sizes larger than 20K are discouraged when copying to tape because the records get very long and are less reliable.

The `dd` command will not handle multiple tapes, so if the partition size exceeds one tape, use multiple `dd` commands. For example, enter:

```
$ dd if=/dev/rxylg of=/dev/rst0 bs=10b count=10000
```

Change tapes, then enter:

```
$ dd if=/dev/rxylg skip=10000 of=/dev/rst0 bs=10b count=10000
```

If you block at 20 blocks per record, you can generally fit approximately 50000 blocks on a standard 2400 foot tape (at 1600 BPI).

For the sake of speed, make the block size a multiple (or factor) of the number of blocks per track. Thus, a block size of 16 is optimal for a device with 32 blocks per track.

Buffer Cache Size

Tuning the SunOS Buffer Cache Size

To utilize raw devices fully, you must adjust the size of the ORACLE buffer cache and, if memory is limited, the SunOS buffer cache.

The SunOS buffer cache is provided by the operating system and is used to hold blocks of file system data, in memory, which are being transferred from memory to disk or vice versa.

This is a configurable parameter on most SunOS systems (NNBUF).

When moving to raw devices, you no longer need a large SunOS buffer cache for the database buffers. To offset this, you must increase the size of the ORACLE buffer cache. If the amount of memory on the system is limited, you should make a corresponding decrease in the SunOS buffer cache size.

Determining cache hit ratio

A "cache hit ratio" tells you what percentage of the time a block is found in the cache and disk I/O is unnecessary. This ratio is determined by the formula:

$$(\# \text{ logical reads} - \# \text{ physical reads}) / \# \text{ logical reads}$$

Here, the number of *logical reads* is the sum of **CONSISTENT_GETS** plus **DB_BLOCK_GETS**.

The ORACLE cache hit ratio is determined from the SQL*DBA **monitor** and **statistics** displays. (It can also be determined from the **user** display for individual processes.)

Helpful System V commands are:

sar -b	reports the SunOS buffer cache activity
sar -w	reports the SunOS swapping activity
sar -u	reports CPU utilization
sar -r	reports memory utilization
sar -p	reports the SunOS paging activity

By running a typical application mix of SunOS programs and ORACLE queries, you can study the **sar** information to determine which buffer caches should be increased or decreased. See your system administrator's guide for a complete description of the **sar(1)** utility.

Adjusting cache size

Here are the general rules for adjusting cache size:

- Increase ORACLE cache size as long as the cache hit ratio goes up.
- Increase the SunOS cache size.
- Decrease the cache sizes if the swapping/paging activity becomes high.

Resource Limits and Disk Quotas

Resource limits under SunOS are inherited from the parent process. (See `getrlimit(2)` in the operating system documentation.) This means that the ORACLE RDBMS shadow process that executes on behalf of any user process will be constrained by these limits. As shipped, SunOS default resource limits are sufficiently high so as not to impede the running of any ORACLE RDBMS shadow or background process. However, if these limits are administratively lowered, the ability of users to make full use of the ORACLE RDBMS system could be impaired. This issue should be discussed with your SunOS system manager.

Disk quotas established for the ORACLE `dba` user-id can also hinder the operation of the ORACLE RDBMS system. Any such quotas should be established only after discussion between the ORACLE DBA and the SunOS system manager.



PART 3: PRODUCTS

This part of the *ICG* contains SunOS-specific information about the Oracle products that may be available on your system. Part 3 is divided into the following sections:

- Core Products
- Oracle Programmatic Interfaces
- SQL*Net V1 Products
- SQL*Net V2 Products
- Office Automation Products
- Additions

CHAPTER

7

ORACLE CORE PRODUCT UPDATES

This chapter presents system-specific information that complements the chapter for each Oracle core product in the *Oracle Tools for UNIX Administrator's Reference Guide*. The Oracle core products include:

- Oracle*Terminal
- SQL*Forms V3.0
- SQL*Menu V5.0
- SQL*Report
- SQL*ReportWriter V1.1

If a section for one of these products is *not* included in this chapter, no additional operating-system specific information is necessary.

SQL*Plus

The **glogin.sql** setup file

The **glogin.sql** file is located in the **\$ORACLE_HOME/sqlplus/admin** directory.

Using the **SPOOL** command

LPDEST is the environment variable used to set the default printer. You must set the **LPDEST** variable to use the **SPOOL** command to send reports directly to a printer other than the default printer.

Using the **SPOOL** command

PRINTER is the environment variable used to set the default printer. You must set the **PRINTER** variable to use the **SPOOL** command to send reports directly to a printer other than the default printer.

The default filename extension for files generated by the **SPOOL** command under SunOS is **.lst**. You can change the extension by specifying a spool file that contains a period (.). For example:

```
SQL> spool query.lis
```

SQL*Forms V3.0

Restrictions

Select Columns

The Select Columns button from the Block/Default window does not function properly. The user will only see the last column of the named table.

Monitors

If you use any monitor other than Sun monitors, invoke `xinit` with the `-dpi 75` option. This assures that SQL*Forms V3 will use 75 dots-per-inch fonts. Fonts that are 100 dpi do not scale properly on any Sun monitors except those with the highest resolution. This excludes Sun color and most Sun monochrome monitors. When you use this option, shut down the X server and restart it.

8-Bit Input

In ORACLE V6.0 releases earlier than 6.0.27, in order to enter 8-bit input (foreign languages) to SQL*Forms applications, you had to enter one of the following commands either `stty -istrip` or `stty pass8` before invoking SQL*Forms:

```
stty pass8
or
stty -istrip
```

These commands tell the terminal driver not to strip the eighth bit. As of V6.0.27, you no longer need to do this, because the terminal driver is set to accept 8-bit input automatically when you invoke SQL*Forms.

Note: Generally, the 8-bit mode will work only through hard-wired ports. The `rlogin` and `telnet` facilities will strip the eighth bit. A work-around for `rlogin` is to enter `rlogin -8`. The `-8` is a flag that turns 8-bit support on.

On BSD-based UNIX systems (or dual universe systems running in the `ucb` universe) problems can occur when you run a VT200 series terminal in VT100 emulation mode with 8-bit communications, and then enter an ORACLE utility that uses the ORACLE VT200 or VT220 `crt` definition.

The affected utilities include SQL*Forms. The VT2XX `crt` definitions for this utility automatically switch your terminal from whatever control mode it was in initially, to "VT200 7-bit controls." This mode enables the `crt` definition to access the "F" keys at the top of the keyboard (which the VT100 `crt` definition does not use).

The problem arises if your terminal's communications mode is still set to 8 bits; to run in "VT200 7-bit controls" mode, the terminal's communications mode must also be set to 7 bits. Your screen will be corrupted if the communications mode is set to 8 bits. While ORACLE crt definitions do change the terminal's control mode, they do not change the terminal's communications mode. This problem concerns only the machine you are directly logged on to, not the machine your application is running on. Here is an example in which "N" is a machine which does **not** interpret the terminal modes correctly, and "D" is a machine which **does** interpret them correctly,

- If you are logged on to N and you try to run SQL*Forms over the network on D, the screen will be corrupted.
- If, however, you are logged on to D and you run SQL*Forms over the network on N, your screen will be fine.

To avoid problems with VT2XX crt definitions, manually set the terminal to 7-bit communications mode whenever you invoke SQL*Forms.

Please note also that the VT2XX crt definitions reset the terminal to VT100 mode when you exit from ORACLE utilities. For example, if your terminal is running SQL*Forms in VT200 mode, the crt definition will change your terminal to VT100 mode when you exit from SQL*Forms. To avoid this, simply change the VT2XX crt definition. For details, please see the manual for your VT220 terminal.

Restrictions

8-Bit Input

On BSD-based UNIX systems (or dual universe systems running in the `uch` universe) problems can occur when you run a VT200 series terminal in VT100 emulation mode with 8-bit communications, and then enter an ORACLE utility that uses the ORACLE VT200 or VT220 `crt` definition.

The affected utilities include SQL*Menu. The VT2XX `crt` definitions for SQL*Menu switch your terminal from whatever control mode it was in initially, to "VT200 7-bit controls." This mode enables the `crt` definition to access the "F" keys at the top of the keyboard (which the VT100 `crt` definition does not use).

The problem arises if your terminal's communications mode is still set to 8 bits; to run in "VT200 7-bit controls" mode, the terminal's communications mode must also be set to 7 bits. Your screen will be corrupted if the communications mode is set to 8 bits. While ORACLE `crt` definitions do change the terminal's control mode, they do not change the terminal's communications mode.

This problem concerns only the machine you are directly logged on to, not the machine on which your application is running. Here is an example in which "N" is a machine which does **not** interpret the terminal modes correctly, and "D" is a machine which **does** interpret them correctly,

- If you are logged on to N and you try to run SQL*Menu over the network on D, the screen will be corrupted.
- If, however, you are logged on to D and you run SQL*Menu over the network on N, your screen will be fine.

To avoid problems with VT2XX `crt` definitions, manually set the terminal to 7-bit communications mode whenever you invoke SQL*Menu.

Please note also that the VT2XX `crt` definitions reset the terminal to VT100 mode when you exit from ORACLE utilities. For example, if your terminal is running SQL*Forms in VT200 mode, the `crt` definition will change your terminal to VT100 mode when you exit from SQL*Forms. To avoid this, simply change the VT2XX `crt` definition. For details, please see the manual for your VT220 terminal.

SQL*ReportWriter V1.1

Administration

Notes

Specifying printers
and queues

The script `$ORACLE_HOME/sqlreport/admin/printer/spoolcmd` is called when you send a file to the printer. Two versions of this script are supplied. The DBA can specify which version to use by moving to the `$ORACLE_HOME/sqlreport/admin/printer` directory and entering the command below.

For the lp command:

```
$ ln spoollp.sh spoolcmd
```

Specifying the
temporary file location

The environment variable `SRW_TMP` defines where temporary files go.

The default setting is `/tmp`.

Restrictions

Shelltool User Exits

SQL*ReportWriter functions properly only in **shelltool** windows.

CHAPTER

8

ORACLE PRECOMPILERS PRODUCT UPDATES

This chapter presents port-specific information that complements the chapter for each product in the *Oracle Tools for UNIX Administrator's Reference Guide*. It is organized as follows:

- Pro*Ada
- Pro*C
- Pro*COBOL
- Pro*FORTRAN

If a section for one of these products is *not* included in this chapter, no additional operating system-specific information is necessary.

Pro*Ada

Administering Pro*Ada

System Libraries

The Oracle precompiler libraries are named so that they can be easily moved to `/usr/lib` if desired. This simplifies the loading step somewhat.

Installing programs and libraries in system directories often requires a system administrator with super-user authority. Control and updating may thus become somewhat more difficult due to loss of some direct control on the part of the database administrator.

The `/usr` directory must be writeable to do this. It may be read-only by default. If this is the case, have your system manager perform the move.

Linking instructions

The library directory for Pro*Ada (`$ORACLE_HOME/proada/lib`) contains the file `proada.mk`. During the installation, this file is linked to the `demo` subdirectory. In the `demo` subdirectory, you can use the `make` command and the `proada.mk` file to compile and link the demonstration program.

Note also that the character array for the structure `hst` in `c/demo/sample.c` should be 512 instead of the standard 256.

Pro*C

Administering Pro*C

System Libraries

The SunOS system provides three compilers for C programs: a BSD 4.2 compiler, `/bin/cc` and a System V compiler, `/usr/5bin/cc`. You must use the System V compiler to link ORACLE programs.

The ORACLE precompiler libraries are named so that they can be easily moved to `/usr/lib` if desired. This simplifies the loading step somewhat. An example of a command line for linking a program then changes from:

```
$ cc -o foo.c $ORACLE_HOME/rdbms/lib/libsql.a
```

to a simpler:

```
$ cc -o foo.c -lsql
```

Note: You must use the version of `cc` in `/usr/5bin`. Either place `/usr/5bin` in your `PATH` before `/bin`, or alias `cc` to `/usr/5bin/cc`, or specify the full pathname `/usr/5bin/cc`. The example above assumes one of the first two methods.

Installing programs and libraries in system directories often requires a system administrator with super-user authority. Control and updating may thus become somewhat more difficult due to loss of some direct control on the part of the database administrator.

The `/usr` directory must be writeable to do this. It may be read-only by default. If this is the case, have your SunOS system manager perform the move.

Pro*COBOL

Administering Pro*COBOL

System Libraries

The precompiler libraries are named so that they can be easily moved to `/usr/lib` if desired. This simplifies the loading step somewhat.

Installing programs and libraries in system directories often requires a system administrator with super-user authority. Control and updating may thus become somewhat more difficult due to loss of some direct control on the part of the database administrator.

Note: The `/usr` directory must be writeable to do this. It may be read-only by default. If this is the case have your system manager perform the move.

Setting the COBOL Environment

The Pro*COBOL precompiler requires you to set an environment variable, `COBDIR`, to the home directory of the compiler. Also, add `$COBDIR/bin` to the `PATH` variable.

Using Pro*COBOL

Sample Sun COBOL Program

A sample COBOL program, `samplecob.pco`, is available in your `$ORACLE_HOME/procob/demo` directory. Before you can run this COBOL program, you must build your demonstration database using the `demobld.sql` file, as discussed in "Installing the SQL*Plus Demonstrations Manually" in the SQL*Plus chapter.

1. Make sure the `ORACLE_HOME` and `COBDIR` environment variables are set in your environment. Also, `$COBDIR` and any other directory location containing COBOL compiler executables should be part of your `PATH` environment variable.
2. Move to the `$ORACLE_HOME/procob/demo` directory:

```
$ cd $ORACLE_HOME/procob/demo
```

Then enter the command:

```
$ make -f procob.mk samplecob.gnt
```

3. To execute this sample program, enter:

```
$ rtsora samplecob.gnt
```

You can now add additional employees to the demonstration database, and display the employees of each department.

Note: If your COBOL program uses DISPLAY statements, you may need to set the environment variable **TERMCAP** to **/usr/lib/cobol/termcap**.

Note: It is not possible to write COBOL user exits using this COBOL compiler.

Pro*FORTRAN

Administering Pro*FORTRAN

System Libraries

The Oracle precompiler libraries are named so that they can be easily moved to `/usr/lib` if desired. This simplifies the loading step somewhat.

Installing programs and libraries in system directories often requires a system administrator with super-user authority. Control and updating may thus become somewhat more difficult due to loss of some direct control on the part of the database administrator.

The `/usr` directory must be writeable to do this. It may be read-only by default. If this is the case, have your system manager perform the move.

Using Pro*FORTRAN

Sample FORTRAN Program

A sample FORTRAN program, `samplef.pfo`, is available in your `$ORACLE_HOME/profor/demo` directory. Before this sample FORTRAN program can be run, you must build your demonstration database using the `demobld.sql` file as discussed in the SQL*Plus chapter of this manual.

To run the **profor** utility on this sample FORTRAN program, enter:

```
$ make -f profor.mk -w0,dba samplef
```

To execute the sample FORTRAN program, enter:

```
$ samplef
```

You can now add additional employees to the demonstration database.

Building Pro*FORTRAN Programs

To precompile, compile, and link a program named **userprog.pfo**, use the command

```
$ make -f profor.mk userprog
```

Note that if your program depends upon non-ORACLE libraries, you may have to alter **profor.mk** to include them.

Using the **FORMAT**
option with
FORTTRAN

The precompiler option **FORMAT** specifies the format of input lines for **FORTTRAN**. If you specify **FORMAT=ANSI**, the default, columns 1 to 6 contain an optional sequence number, column 7 indicates comments or continuation lines, paragraph names begin in columns 8 to 11, and statements begin in columns 12 to 72.

If you specify **FORMAT=TERMINAL**, columns 1 to 6 are dropped, making column 7 the leftmost column.

9

SQL*NET V1 PRODUCT UPDATES

This chapter presents port-specific information that complements the chapter for each product in the *Oracle Tools for UNIX Administrator's Guide*. It is organized as follows:

- SQL*Net V1 Overview
- SQL*Net TCP/IP
- SQL*Net Async

If a section for one of these products is *not* included in this chapter, no additional operating-system specific information is necessary.

SQL*Net V1 Overview

Linking Specific Drivers to Executables

The **genosntab** script (described briefly below) determines which SQL*Net V1 or two-task drivers are to be linked to ORACLE executables or precompiled programs. This script exists in the **\$ORACLE_HOME/bin** directory.

The syntax for generating **osntab** is:

```
genosntab [-u | [-n driver_symbol [driver_symbol]] ...] > osntab.c
```

Note: If you do not specify any driver (that is, you do not specify **-n driver_symbol**), all available drivers will be installed.

- are used in a single-system environment
- may be used locally, to connect multiple databases on one machine
- are provided as a standard part of the ORACLE RDBMS

Networking drivers (such as SQL*Net TCP/IP):

- are only used in a network environment (for communication between ORACLE systems on different machines)
- are purchased as separate products (i.e., as various SQL*Net V1 protocols)
- are installed and configured separately

Single system environment

Using the default configuration:

Optimizing for local tasks:

SQL*Net TCP/IP Dispatcher Support

Currently, the dispatchers only support the SQL*Net TCP/IP protocol. This means that if any clients wish to connect to MCS, they must be linked with the SQL*Net TCP/IP driver. Refer to the SQL*Net Chapter in the *TARG* for more information.

orasrv

The **orasrv** listener process must be running in order for the dispatchers and servers to function. By default, **orasrv** will connect clients to MCS. However, the **tcpctl** and **tcputl** utilities have been modified to tell **orasrv** whether to connect clients to MCS (via the Oracle dispatcher) or regular SCS (Single-Client Server). Use the command line option **mcs** to tell **orasrv** to toggle the default option.

```
$ tcpctl mcs
```

```
tcputl: checking user permission...  
tcputl: MCS mode is DISABLED
```

```
$ tcpctl mcs
```

```
tcputl: checking user permission...  
tcputl: MCS mode is ENABLED
```

Connect String

Although the basic format of the connect string has not been changed, this release does allow the user to specify whether to connect to Oracle MCS or regular SCS. To override the default to which **orasrv** has been set, add the suffix **-mcs** or **-scs** to the SID in the connect string. For example, if **orasrv** has been told to connect clients to MCS and the DBA wishes to use SCS, the DBA can use the connect string to do so:

```
$ sqlplus system/manager@T:machine:N-scs
```

This will connect the user to Oracle SCS for the N database located on the remote machine. Note that this feature may change or be obsoleted in the future.

CHAPTER

10 SQL*NET V2 PRODUCT UPDATES

This chapter presents port-specific information that complements the chapter for each product in the *Oracle Tools for UNIX Administrator's Reference Guide*. It is organized as follows:

- DECnet Adapter
- SPX/IPX Adapter
- TCP/IP Adapter

If a section for one of these products is *not* included in this chapter, no additional operating-system specific information is necessary.

Installing SQL*Net V2

Post-Installation Step

Be sure to make the TNS listener process setuid **root**. Do this by entering the following commands as **root**:

```
# cd $ORACLE_HOME/bin
# chmod 4751 tnslnsr
```

SPX/IPX Adapter

Post-Installation Step

Be sure to make the SPX broadcast daemon setuid as **root**. Do this by entering the following commands as **root**:

```
# cd $ORACLE_HOME/bin
# chmod 4751 ntisbasm
```


CHAPTER

11

OFFICE AUTOMATION PRODUCT UPDATES

This chapter presents port-specific information that complements the chapter for each product in the *Oracle for UNIX Product Guide*. It is organized as follows:

- Oracle*Mail

Installation Notes

If you are upgrading Oracle*Mail: If Oracle*Mail has been previously installed on a version of the ORACLE RDBMS within this major release (i.e., 6.0 or 7.0), and the install script was already run as the root user, go directly to the next step.

Administration Notes

Usage Notes

Using Your Operating System Editor

You can use your native operating system editor in Oracle*Mail by setting the environment variable **ORAMAIL_EDITOR** to its executable. The following examples show how to make the **vi** editor accessible from Oracle*Mail.

- **For Bourne shell users:**

```
$ ORAMAIL_EDITOR=/usr/ucb/vi
$ export ORAMAIL_EDITOR
```

- **For C shell users:**

```
% setenv ORAMAIL_EDITOR /usr/ucb/vi
```

Compiling and Linking API Applications

To compile and link an API application, complete the following steps:

1. **Compile using the Oracle*Mail API headers mhdef and mheno:**

```
$ /usr/5bin/cc -O -I$ORACLE_HOME/mail/api/lib -c main.c
```

2. **Link using the Oracle*Mail API library libmh.a**

```
$ /usr/5bin/cc -O -Bstatic -o main main.o \
    $ORACLE_HOME/rdbms/lib/libora.a \
    $ORACLE_HOME/rdbms/lib/libucb.a \
    -lc /lib/libc.a
```

CHAPTER

12

ORACLE ADDITIONS PRODUCT UPDATES

This chapter presents port-specific information that complements the chapter for each product in the *Oracle for UNIX Product Guide*. It is organized as follows:

- Multilingual Option
- PL/SQL
- SQL*Loader
- Oracle XA Library

If a section for one of these products is *not* included in this chapter, no additional operating-system specific information is necessary.

Oracle XA Library

For a complete discussion of the Oracle XA Library, refer to the Oracle XA chapter in the *ORACLE7 Server for UNIX Administrator's Reference Guide*.

Required public information

As a resource manager, Oracle is required to publish the following information:

- Libraries needed to link applications using Oracle XA are:
 - `$ORACLE_HOME/lib/libxa.a`
 - `$ORACLE_HOME/lib/libsql.a`
 - `$ORACLE_HOME/lib/osntab.o`
 - `$ORACLE_HOME/lib/libsqlnet.a`
 - `$ORACLE_HOME/lib/libora.a`
 - `$ORACLE_HOME/lib/libcore.a`

Note: Refer to the `Readme.doc` file for your particular machine to obtain any changes to library filenames.

APPENDIX

A

UNIX SUPPLEMENT TO ORACLE7 SERVER DOCUMENTATION

This appendix is designed to be used as a supplement to the following ORACLE7 Server documents:

- Appendix E of the *Administrator's Guide*
- Appendix B of the *Concepts Guide*
- Appendix F of the *Application Developer's Guide*
- Appendix D of the *Utilities User's Guide*
- Appendix C of the *Migration Guide*
- Appendix D of the *SQL Language Reference Manual*

The above listed appendices contain listings of operating system specific information references. The following lists appear in the same order as those lists.

This Appendix is not intended for use on its own. It is a supplement, and contains mostly references to specific topics covered in your ORACLE for UNIX documentation set.

Any references to chapters in this list refer to chapters in the present book, unless otherwise noted. The *ORACLE7 for SunOS Installation & Configuration Guide* is referred to in this document as the ICG.

The *ORACLE7 Server for UNIX Administrator's Reference Guide* is referred to throughout this chapter as the *SARG*.

The *Oracle Tools for UNIX Administrator's Reference Guide* is referred to throughout this chapter as the *TARG*.

Administrator's Guide

Administrator's OS account	DBA account users must be members of the <i>dba</i> group, defined in <i>/etc/group</i> .						
Archiving mode	See the "Log Archiving" chapter of the <i>SARG</i> for a complete discussion of log archiving. <table><tr><td>initial mode</td><td>The <code>log_archive_start</code> parameter is "false" by default, which means the archiver is off until this value is changed by editing the <i>init.ora</i> file.</td></tr><tr><td>enabling auto</td><td>This is a valid option on SunOS.</td></tr><tr><td>specific steps</td><td>See the "Backing Up and Recovering a Database" chapter of the <i>SARG</i>.</td></tr></table>	initial mode	The <code>log_archive_start</code> parameter is "false" by default, which means the archiver is off until this value is changed by editing the <i>init.ora</i> file.	enabling auto	This is a valid option on SunOS.	specific steps	See the "Backing Up and Recovering a Database" chapter of the <i>SARG</i> .
initial mode	The <code>log_archive_start</code> parameter is "false" by default, which means the archiver is off until this value is changed by editing the <i>init.ora</i> file.						
enabling auto	This is a valid option on SunOS.						
specific steps	See the "Backing Up and Recovering a Database" chapter of the <i>SARG</i> .						
Auditing	 <table><tr><td>creation of views</td><td>On SunOS, the ORACLE database and its data dictionary are not created with the audit trail views. You must run the <i>audit.sql</i> script.</td></tr><tr><td>directing records</td><td>Operating system audits are not currently used with ORACLE for SunOS.</td></tr><tr><td>O/S audits</td><td>Operating system audits are not currently used with ORACLE for SunOS.</td></tr></table>	creation of views	On SunOS, the ORACLE database and its data dictionary are not created with the audit trail views. You must run the <i>audit.sql</i> script.	directing records	Operating system audits are not currently used with ORACLE for SunOS.	O/S audits	Operating system audits are not currently used with ORACLE for SunOS.
creation of views	On SunOS, the ORACLE database and its data dictionary are not created with the audit trail views. You must run the <i>audit.sql</i> script.						
directing records	Operating system audits are not currently used with ORACLE for SunOS.						
O/S audits	Operating system audits are not currently used with ORACLE for SunOS.						
Authentication through OS	Users are authenticated according to their membership in groups as defined in the <i>etc/group</i> file. See the security section in the "Administering ORACLE" chapter of the <i>SARG</i> for more information on user authentication.						
Backups	See the "Backing Up and Recovering a Database" chapter in the <i>SARG</i> for information on all of your backup options.						
Character Sets	Available characters sets are listed in the "Multilingual Option" chapter in the <i>SARG</i> .						

Clusters, estimating size

Use the following figures for calculating cluster size the *Administrator's Guide*:

- fixed header size = 68 bytes
- variable transaction header = 24 (*INITTRANS* value for the table)
- row directory = 4 bytes per row of a clustered table

Control files

For more information on control files, see the "Recommendations for Configuring your Database" section of the "Configuring your ORACLE System" chapter.

default name	The default name of the control file created at installation is <code>\$ORACLE_HOME/dbs/cntrl1sid.ctl</code> .
specifying names	You can make copies of the default control file, with names such as <code>pathname/cntrl2sid.ctl</code> .
maximum size	2500*(minimum physical block size at the device level). See the "Key to ORACLE on SunOS" chapter for the block size of your UNIX platform.

Connect strings

For information on specifying a SQL*Net connect string, see the SQL*Net chapters in the *TARG*.

Creating a database

during installation	You can choose the COMPLETE SOFTWARE/DATABASE FRESH INSTALLATION option in the Installer to create a default database during the installation process. Alternatively, you can choose the Create New Database Objects option in the Installer to create a database after the initial software installation. See the "Installing Oracle Products" chapter for installation information.
OS specific steps	See the "Creating a Complex Database" chapter of the <i>SARG</i> for information on creating databases on UNIX.

Data files

creating and naming	The installation process creates one default database file for each of the following tablespaces: SYSTEM,
---------------------	---

TEMP, USERS, RBS, and TOOLS in the pattern **\$ORACLE_HOME/dbs/tablespace.dbf**. For information on adding datafiles, see the "Administering ORACLE" chapter in the *SARG*.

limit on number ORACLE for SunOS supports 1022 concurrently open files.

Database

block size The default value of the **init.ora** parameter **db_block_size** is listed in the "Initialization Parameters" section provided later. The current value on your system can be found by using the SQL*DBA command **show parameters**.

database limits For the equation given in the *Administrator's Guide*, use the following figures: database file size = 5Mb, database files= 255 or the value of the **db_files** parameter in the **init.ora** file, **instances**=255, **locks**=100 (**dml_locks** parameters value), **MAXEXTENTS**=121 for a 2k block size, 249 for a 4k block size, and 505 for an 8k block size, redo log files =255.

default directory Data files are located in **\$ORACLE_HOME/dbs** by default.

Database links

connect string For information on SQL*Net connect strings, see the SQL*Net chapters in the *TARG*.

Dedicated server, requesting

You must specify **SERVER=DEDICATED** at the end of the **CONNECT_DATA** section of your connect descriptor, for example:

```
(DESCRIPTION = (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = prodhost)
                (PORT = tnslsnr))
(CONNECT_DATA =
  (SID = PRODDb)
  (SERVER = DEDICATED)))
```

For SunOS-specific information on the ADDRESS = portion of the connect discriptor, see the SQL*Net V2 chapters in the TARG.

Disk use, monitoring

See the "Tuning" section of the "ORACLE System Administration" chapter for information on monitoring your system. See also the "Monitoring and Tuning your System" chapter of the SARG.

Dispatcher process

connect string	The MTS_ initialization parameters listed in the "Parameter Files and Initialization" appendix of the <i>Administrator's Guide</i> must be set. See the "Initialization Parameters" section provided later for the default SunOS values.
----------------	--

Distributed transaction recovery Files

Single process instances are not generally used on SunOS.

created initially	If you choose to have the Installer create a default database for you, you will be prompted for data file names during the installation. If you create your own database, you can specify filenames in the create database command, or use the default (\$ORACLE_HOME/dbs/tablespacesid.dbf).
number allowed	ORACLE for SunOS supports 1022 concurrently open files.
specifying names	By default, control files are named \$ORACLE_HOME/dbs/ctrl1sid.ctl.

Hardware, evaluating

See the "Key to ORACLE" chapter for hardware supported on SunOS.

Installing products

See the "Installing Oracle Products" chapter for complete information about installation.

Instance

check identifier	A database instance is identified by the value of ORACLE_SID . Use the SQL*DBA command show instance to display.
startup	Automatic instance startup is controlled by the dbstart shell script, described in the "Completing Your Installation" section of the "Installing Oracle Products" chapter. Startup of remote instances is dependent on your SQL*Net environment and whether or not you have enabled remote connections with the orapasswd utility. See the SQL*Net chapters in the <i>TARG</i> for more information. See also the "Administering ORACLE" chapter of the <i>SARG</i> for a discussion of orapasswd .

INTERNAL

and non-secure connections	Use the orapasswd utility to disable remote connections. You can specify this during installation.
connecting as	You must be a member of the DBA group to connect as INTERNAL.
password for	The Installer will prompt you for a password for INTERNAL privileged accounts.

Latch contention threshold

See the "Tuning" section of the "ORACLE System Administration" chapter for a discussion of latches.

Listener process

For information on TNS listener processes, see the SQL*Net V2 chapters in the *TARG*.

maxlogfiles, maxlogmembers

The maximum limit of the **maxlogfiles** parameter is 255. The maximum limit of the **maxlogmembers** parameter is 5.

Migration from V6 to V7

See the "Upgrading and Migrating Your System" chapter for complete information on all your migration options.

Multi-Threaded Server

For information on MTS connect strings, see the SQL*Net V2 chapters in the *TARG*.

National Language Support (NLS)

For information on National Language Support, see the “Multilingual Option” chapter in the *SARG*.

error message file	Look for National Language Support errors in <i>product/mesg/faclang.msg</i> .
NLS_LANG	To set the parameter from the Bourne or Korn shell enter the following commands from your system prompt: <pre>\$ NLS_LANG = language_territory.charset \$ export NLS_LANG</pre> From the C shell, enter: <pre>% setenv NLS_LANG language_territory.charset</pre>
NLS_LANGUAGE	For a list of languages, territories, and character sets, see the “Multilingual Option” chapter in the <i>SARG</i> .
NLS_SORT	For a list of supported sort sequences, see the “Multilingual Option” chapter in the <i>SARG</i> .
NLS_TERRITORY	For a list of languages, territories, and character sets, see the “Multilingual Option” chapter in the <i>SARG</i> .

Parallel Server

The Parallel Server option is not currently supported on SunOS.

Parameter files

case sensitive	UNIX is case sensitive. All values listed for parameters should be entered exactly as they appear in your ORACLE for UNIX documentation.
creating and editing	The init.ora parameter file is created upon initial installation, and can be edited as a text file on UNIX using a system editor.
default filename	Although the <i>sid</i> is listed in this file name, we refer to it throughout the <i>ICG</i> as simply init.ora .
format	See the <i>SARG</i> for the default init.ora file supplied with your distribution.
global cache values	The Parallel Server option is not currently supported on SunOS.

	location	The init.ora file is located in \$ORACLE_HOME/dbs by default.
	O/S parameters	See the "Initialization Parameters" section below.
Recovery		See the "Operations Planning" section of the "Backing Up and Recovering a Database" chapter in the <i>SARG</i> for information about recovery.
Redo log files		See the "Backing Up and Recovering a Database" chapter of the <i>SARG</i> for more information.
	destination search path	This is denoted by the value of the \$ORACLE_PATH variable.
	default location	The redo log files are located by default in \$ORACLE_HOME/dbs .
	default size	The default value of the log_size variable is 500k.
	limited by log_files	The default value of log_files is 255.
REFSNAP utility		The refresh utility is located in \$ORACLE_HOME/bin/refsnap .
Roles		See the "Groups and Security" section of the "Administering ORACLE" chapter of the <i>SARG</i> for information on groups and roles. For the role specification syntax given in the <i>Administrator's Guide</i> , <ID> refers to your <i>sid</i> . The role should be specified in lower case.
		See also the section "Security and Remote PC Users" for a discussion of the orapasswd utility.
	OSOPER and OSDBA	You can map UNIX groups to ORACLE roles in the /etc/group file.
	using the O/S for security	UNIX checks /etc/group to verify that the user logging is a member of the appropriate group, and which privileges that user has been assigned.
Security		For more information on security, see the "Security Considerations" section in the "Administering ORACLE" chapter in the <i>SARG</i> .

SQL scripts, names and locations

All .sql scripts must be run as the **sys** user. With the exception of **\$ORACLE_HOME/dbs/sql.bsq**, all .sql scripts are located in **\$ORACLE_HOME/rdbms/admin**.

to run initially

If you are using the procedural option, you must run **catproc.sql**, located in **\$ORACLE_HOME/rdbms/admin**.

You must run **catalog.sql** after creating a database with the **create database** command.

Storage parameters, defaults

See the "SQL Language Reference Manual" section for the default and maximum values of storage parameters.

SQL*DBA

installing

There are no special considerations for installing SQL*DBA on SunOS.

setup

To run SQL*DBA in full screen mode (the default), you must set the **TK2DEV** environment variable to define your terminal definition. Available **TK2DEV** values are listed in the "Installing Oracle Products" chapter. The default value is **vt100**.

starting

If you want to run SQL*DBA in line mode, you must specify that when you invoke SQL*DBA:

```
$ sqldba lmode=y
```

By default, SQL*DBA is invoked in full screen mode.

Tables, estimating size

clustered

See the "Clusters" section above.

non-clustered

Use the same values listed in the "Clusters" section for calculating non-clustered tables.

Tablespaces, default storage parameters

See the "SQL Language Reference Manual" section for the default and maximum values of storage parameters.

Trace files

Trace files appear in the following format: *processname_unixpid.trc*, where *processname* is the ORACLE process name (*ora*, *dbwr*, *pmon*, *lgwr*, *smon*, *reco*, *arch*, *d000*, or *s000*), and *unixpid* is the UNIX process number. The *.trc* file extension is appended to all trace files. Trace and Alert files are discussed in the "Administering ORACLE" chapter of the *SARG*.

Tuning information

See the "Tuning" section of the "ORACLE System Administration" chapter for information specific to SunOS. For general UNIX administration information, see the *SARG*.

Initialization parameters

Below are listed the default values of these parameters.

BACKGROUND_DUMP_DEST	\$ORACLE_HOME/rdbms/log
COMMIT_POINT_STRENGTH	1
CONTROL_FILES	0
DB_BLOCK_SIZE	2048
DB_FILES	30
DB_FILE_MULTIBLOCK_READ_COUNT	4 (Range of 1-20)
DB_FILE_SIMULTANEOUS_WRITES	4
DISTRIBUTED_TRANSACTIONS	6
INIT_SQL_FILES	\$ORACLE_HOME/dbs/sql.bsq
INSTANCE_NUMBER	0
LOG_ARCHIVE_BUFFER_SIZE	127
LOG_ARCHIVE_BUFFERS	4
LOG_ARCHIVE_DEST	\$ORACLE_HOME/dbs/arch
LOG_ARCHIVE_FORMAT	_%.dbf
LOG_BUFFER	4x maximum block size
LOG_CHECKPOINT_INTERVAL	500x(max block size/min. block size)
LOG_SMALL_ENTRY_MAX_SIZE	800
MTS_MAX_DISPATCHERS	5
MTS_MAX_SERVERS	20
MTS_SERVERS	N/A
NLS_LANGUAGE	american
NLS_TERRITORY	america
OPEN_CURSORS	50 per process
OS_AUTHENT_PREFIX	ops\$
PROCESSES	25
SHARED_POOL_SIZE	3.5 MB
SORT_AREA_SIZE	65536
SORT_SPACEMAP_SIZE	512
TEMPORARY_TABLE_LOCKS	N/A
TRANSACTIONS_PER_ROLLBACK_SEGMENT	30
USER_DUMP_DEST	\$ORACLE_HOME/rdbms/log

Concepts Guide

Authentication through OS

Users are authenticated according to their membership in groups as defined in the `etc/group` file. See the "Network Security" section of the "Administering ORACLE" chapter of the *SARG* for information on user authentication on UNIX.

Background processes

For a description of background processes, see the "ORACLE System Architecture" chapter in the *SARG*.

creating

Background processes are created automatically when an instance is started. For more information on setting up server processes, see the "SQL*Net V1 Overview" chapter in the *TARG*.

Character Sets

available

See the "Multilingual Option" chapter in the *SARG* for information on available NLS character sets.

Client/Server communication

For information on communications links, see the SQL*Net chapters in the *TARG*.

Communications software

Requirements for communications software are detailed in the "Key to ORACLE" chapter.

Configuring ORACLE

UNIX supports both multi-threaded and dedicated server configurations.

Database links

connect string and network domain service

See the SQL*Net chapters in the *TARG* for information on specifying connect strings.

remote OS authentication

See the "Network Security" section of the "Administering ORACLE" chapter in the *SARG*.

Dedicated server, requesting

You must specify `SERVER=DEDICATED` at the end of the `CONNECT_DATA` section of your connect descriptor, for example:

```

(DESCRIPTION = (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = prodhost)
                (PORT = tnslnsr))
(CONNECT_DATA =
  (SID = PRODDb)
  (SERVER = DEDICATED)))

```

For SunOS-specific information on the ADDRESS = portion of the connect descriptor, see the SQL*Net V2 chapters in the *TARG*.

**INTERNAL,
prereqs for
connecting as**

You must be a member of the DBA group to obtain the connect internal privilege.

**Program Global
Area (PGA), size of**

Use the disk space and memory charts in the "Key to ORACLE" chapter to determine your memory usage.

Roles

See the "Groups and Security" section of the "Administering ORACLE" chapter in the *SARG* for information on groups and roles. For the role specification syntax given in the *Concepts Guide*, <ID> refers to your *sid*. The role should be specified in lower case.

**Software code
areas, shared**

Shared software code areas are not supported on SunOS.

**SQL*Net,
networking**

For a thorough discussion of SQL*Net and networking, see the SQL*Net chapters in the *TARG*.

Applications Developer's Guide

Parameter files	Default parameter values are listed in the "Administrator's Guide" section. You can change the values by editing the <code>init.ora</code> file.
SQL scripts, names and locations	With the exception of <code>\$ORACLE_HOME/dbs/sql.bsq</code> , all <code>.sql</code> scripts are located in <code>\$ORACLE_HOME/rdbms/admin</code> .
Server message file	Error messages are located in <code>product/mesg/faclang.msg</code> where <i>product</i> is the name of the product, and <i>faclang</i> is the name of the language. For a listing of available languages, see the chapter on Multilingual Option in the <i>SARG</i> .

Migration Guide

Location of convert.ora file

You can find the convert file as `$ORACLE_HOME/dbs/convsid.ora`.

Running the migration utility

See the "Upgrading and Migrating your ORACLE System" chapter for all migration paths.

The steps that accompany the "Using the Migration Utility" section of the *Migration Guide*, and provide the SunOS-specific instructions referred to in that document are located in the "Upgrading and Migrating your ORACLE System" chapter. Please read both sections carefully to make sure you understand the sequence of steps before proceeding with the migration.

Installing ORACLE7

See the "Installing Oracle Products" chapter for complete information on installing ORACLE for SunOS.

Format of ALERT filenames

The file named `alertsid.log` is located in the directory specified by the `init.ora` parameter `background_dump_dest`, whose default is `$ORACLE_HOME/rdbms/log`.

Format of TRACE filenames

Trace files appear in the following format: `processname_unixpid.trc`, where `processname` is the ORACLE process name (`ora`, `dbwr`, `pmon`, `lgwr`, `smon`, `reco`, `arch`, `d000`, or `s000`), and `unixpid` is the UNIX process number. The `.trc` file extension is appended to all trace files. Trace and Alert files are discussed in the "Administering ORACLE" chapter of the *SARG*.

Utilities Guide

General

installing utilities	ORACLE utilities are installed with ORACLE7 Server. There are no special considerations for installing the utilities.
README file	For last minute updates to this release, see <code>\$ORACLE_HOME/rdbms/admin/readmeunix.doc</code> .

SQL*DBA

backups	See the "Backing up and Recovering a Database" chapter of the <i>SARG</i> for complete information on all of your backup options.
default spool file	The default spool file name is <code>sqldba.log</code> .
extra quotes	No extra quotation marks are required for SunOS.
file locations	See the "Backing up and Recovering a Database" for information on specifying archive destination files.
initial default instance	The initial instance is set with the set instance command. If a value is defined for the TWO_TASK environment variable, that becomes the default and will override any setting for ORACLE_SID . If TWO_TASK is not set, then ORACLE_SID will be the default instance.
key mappings	Use of [CTRL]k is supported to display key mappings in SQL*DBA.
max data size	The maximum value is 65535. The default is 20480.
monitors	
content of fields	There are no SunOS-specific considerations for the contents of the monitor fields.
interrupt	[CTRL]c can be used as an interrupt mechanism.
privileges for INTERNAL	You must be a member of the DBA group to obtain the connect internal privilege.
range of width of	

CHARWIDTH	The range of values is 1–65535.
DATEWIDTH	The range of values is 1–65535.
LONGWIDTH	The range of values is 1–65535.
NUMWIDTH	The range of values is 1–65535.
returning from OS	After shelling out of SQL*DBA, type exit at your system prompt to return.
SQL*Net connect strings	See the SQL*Net chapters in the <i>TARG</i> for a discussion of connect strings.

Import/Export

error messages	See the <i>SARG</i> for a complete list of error messages for ORACLE on UNIX.
installing exp views	Use the catexp.sql script, located in \$ORACLE_HOME/rdbms/admin .

SQL*Loader

See the “ORACLE System Architecture” chapter of the *SARG* for information on SQL*Loader on SunOS.

backslash escape	ORACLE for SunOS interprets the backslash as an escape character.
sizes	
INTEGER	The length of this parameter is typically 4 bytes.
FLOAT	The length of this parameter is typically 8 bytes.
DOUBLE	The length of this parameter is typically 8 bytes.
SMALLINT	The length of this parameter is typically 2 bytes.
VARCHAR	This parameter is determined by 2 + length of the VARCHAR field, where 2 is the equivalent of SHORTINT.
VARGRAPHIC	This parameter is determined by 2 + 2(maximum length of the VARGRAPHIC field), where 2 is the equivalent of SHORTINT.
file processing options	See the “ORACLE System Architecture” chapter of the <i>SARG</i> for information on the file processing string.

**CREATE
CONTROLFILE
parameters**

Parameter	Default Value	Maximum Value
maxdatafile	30	1022
maxinstances	63	63
maxlogfiles	16	255
maxlogmembers	2	5
maxloghistory	100	63000

**CREATE
DATABASE
parameters**

Parameter/File Name	Default Value	Maximum Value
datafile	dbf1sid.dbf	n/a
logfile	log1sid.dbf	n/a
maxdatafile	30	1022
maxinstances	63	63
maxlogfiles	16	255
maxlogmembers	2	5
maxhistory	100	63000
character set	us7ascii	n/a

**STORAGE
parameters**

The maximum value for these parameters is any positive integer. There is no numeric limit.

- **initial**
- **next**
- **minextents** (limited by **maxextents** value)
- **pctincrease**
- **optimal**

**ROWID component
lengths**

block	The length of the block component is 8 characters.
file	The length of the file component is 4 characters.

Filenames and locations

All .sql scripts are located in \$ORACLE_HOME/rdbms/admin except sql.bsq, which is located in \$ORACLE_HOME/dbs.

INDEX

SYMBOLS

.sql scripts, run by the Installer, 3 – 27
\$
 in ops\$ login on command line, 1 – 31
 prompt, 1 – 23
%, prompt, 1 – 23
@, with SQL*Plus, 1 – 27

A

AIO. *See* asynchronous I/O
API applications, 11 – 2
APIPATH, default value, 1 – 26
archive monitor, workaround, 1 – 29
archiving
 enabling, 3 – 23
 parameters, 6 – 9
async_write, 6 – 7
asynchronous I/O, 6 – 13
audit trails, A – 3

B

background processes, A – 13

background_dump_dest parameter, default value, A – 12
block size, 1 – 24
 db_block_size parameter, 6 – 9
 specifying, 1 – 24
blocksize, database, vs maxextents, 1 – 25
Bourne shell, prompt, 1 – 23
buffer cache size. *See* cache size

C

C shell, prompt, 1 – 23
cache hit ratio, determining, 6 – 23
cache size, 6 – 23
 adjusting, 6 – 24
character set parameter, default value, A – 19
charwidth, range of values, A – 18
clusters, estimating size, A – 4
COBDIR, 1 – 30, 8 – 4
COBOL. *See* Pro*COBOL
commit_point_strength parameter, default value, A – 12
compatibility, between ORACLE versions, 1 – 32
Compiling API applications, 11 – 2
control files, default names, 2 – 23

- COPY command, workaround, 1 – 33
- coraenv, 3 – 23
- cpu_count, 6 – 8
- cpu_count parameter, default value, A – 12
- create controlfile parameters, default and max values, A – 19
- create database parameters, default and max values, A – 19
- .cshrc file, 3 – 23
- customized files, saving, 4 – 15

D

- data files, default names and sizes, 2 – 23
- database files
 - default specification, 1 – 31
 - size, 1 – 25
- datafile parameter, default value, A – 19
- datewidth, range of values, A – 18
- db_block_compute_checksums, 6 – 6
- db_block_size parameter, 1 – 24, 6 – 9
 - default value, A – 12
- db_block_write_batch, 6 – 7
- db_file_multiblock_read_count, 6 – 7
 - default value, A – 12
- db_file_simultaneous_writes, default value, A – 12
- db_files parameter, default value, A – 12
- db_writers parameter, 6 – 6
- DBS_SIZE
 - default value, 1 – 26
 - set for installation, 5 – 27
- dbshut
 - general description, 3 – 24
 - setup, 3 – 26
- dbstart
 - general description, 3 – 24
 - setup, 3 – 26
- dedicated server, requesting, A – 5, A – 13
- deinstall
 - Interchange, 2 – 30
 - manual, 4 – 4
 - old ORACLE distributions, 4 – 4
 - SQL*Net Asynchronous, 2 – 27

- utility, 4 – 4
 - syntax, 4 – 5
 - using, 4 – 6
- deinstallation, 5 – 27
- directory, location of, 1 – 25
- disk and memory space requirements, 1 – 13
 - descriptions of categories, 1 – 13
 - for applications development products, 1 – 16
 - for extended functionality products, 1 – 14
 - for ORACLE7, 1 – 14
 - for precompilers, 1 – 18
 - use cycle, 1 – 20
 - for standalone products, 1 – 16
 - how to calculate from the charts, 1 – 14
- disk quotas, 6 – 25
- distributed_transactions, default value, A – 12
- dni. *See* SQL*Net DECnet

E

- environmental variables, 1 – 26
- environment variables
 - COBDIR, 8 – 4
 - default values, 1 – 26
 - for installation, 5 – 27
 - NLS_LANG, 5 – 28
 - NO_MAKE, 5 – 28
 - NONAMESERVER, 2 – 28
 - ORACLE
 - APIPATH, 1 – 26
 - default values, 1 – 26
 - ORAMAIL_EDITOR, 11 – 2
 - SRW_TMP, 7 – 6
 - ORACLE_HOME, 5 – 28
 - ORACLE_PATH, 5 – 28
 - ORACLE_SID, 5 – 28
 - TMPDIR, 5 – 30
- errors, shutdown, 1 – 33
- /etc/fstab, 5 – 31
- exit codes, UNIX restriction, 1 – 32
- export, with multiple tapes, 4 – 13
- export upgrade, 4 – 15
 - brief description, 4 – 11

- performing, 4 – 16
- export utility, description, 4 – 13
- export views, installing, A – 18
- extents, maximum number of, 1 – 24

F

- Fast Two-Task driver, workaround, 1 – 34
- file names
 - default extensions in SQL*Plus, 7 – 2
 - length restrictions, 1 – 26
- files
 - customized, saving, 4 – 15
 - database, 1 – 25
 - init.ora, modifying, 3 – 22
 - log, 1 – 25
 - online help, 3 – 4
 - README, 3 – 4
 - size, 1 – 24
- fstab, 5 – 31

G

- glogin.sql command file, 7 – 2

I

- I/O activity, monitoring performance, 6 – 10
- import
 - example of syntax, 1 – 32
 - with multiple tapes, 4 – 13
- import utility, description, 4 – 13
- init.ora file
 - description, 5 – 29
 - limit to concurrent processes, 1 – 28
 - modifying parameters, 3 – 22
 - parameter restrictions, 1 – 32
- init_sql_files parameter, default value, A – 12
- initial parameter, maximum value, A – 19
- initialization parameters, default values, A – 12

- installation, 3 – 2
 - See also* Installer
 - completing, 3 – 22
 - environment variables, 5 – 27
 - logging, 3 – 15
 - network configuration, 5 – 31
 - patches, when to install, 4 – 11
 - preinstallation activities, 2 – 21
 - prompts for tools, 3 – 18
 - restrictions, 1 – 31
 - space for relinking, 5 – 30
 - SQL*Net V2, 10 – 2
- installation checklist, – 2
- installation log file, install.log, 3 – 15
- Installer
 - arranging windows in, 3 – 12
 - commands, 3 – 6
 - decision chart, 3 – 14
 - exiting, 3 – 6
 - function keys, 3 – 7
 - Install Actions menu, 3 – 10
 - installation log file, 3 – 15
 - installation options, 3 – 13
 - interrupting a session, 3 – 6
 - navigation, 3 – 6
 - online help, 3 – 8
 - overview, 3 – 2
 - overview of install actions, 3 – 13
 - Product Installation screen, 3 – 11
 - Pull-down Menu, 3 – 12
 - restarting, 3 – 6
 - sample input screen, 3 – 8
 - starting, 3 – 16
 - terminology, 3 – 5
 - TK2DEV values, 3 – 7
 - troubleshooting, 3 – 17
 - using, 3 – 5
- installing, where to begin, 2 – 2
- instance, A – 6
- instance_number parameter, default value, A – 12
- Interchange
 - deinstalling old versions, 2 – 30

preinstallation, 2 – 30
iostat tool, 6 – 11

K

keyboards, converting types, 1 – 28

L

latch_wait_mult parameter, 6 – 8
latch_wait_time parameter, 6 – 8
latches, 6 – 8
limits, resource, 6 – 25
log file
 parameters, 6 – 9
 size, 1 – 25
log files, default names and sizes, 2 – 23
log_allocation parameter, 6 – 9
log_archive_buffer_size, default value, A – 12
log_archive_buffers parameter, default value, A – 12
log_archive_dest parameter, default value, A – 12
log_archive_format parameter, default value, A – 12
log_archive_start, initial mode, A – 3
log_blocks_during_backup, 6 – 9
log_buffer parameter, 6 – 9
 default value, A – 12
log_checkpoint_interval, 6 – 10
 default value, A – 12
log_files parameter, 6 – 10
log_simultaneous_copies, 6 – 8
log_small_entry_max_size, default value, A – 12
logfile parameter, default value, A – 19
logging your installation, 3 – 15
logical reads, 6 – 23
logical volumes, 6 – 17
.login file, 3 – 23
longwidth, range of values, A – 18
LPDEST, 7 – 2

M

maintenance, of Oracle products, 5 – 27
man pages, 1 – 23
manual deinstall, 4 – 4
maxdatafile parameter, default and max value, A – 19
maxdatafiles, UNIX limit, 1 – 32
maxextents, database blocksize vs, 1 – 25
maxextents value, 1 – 25
maxinstances parameter, default and max value, A – 19
maxlogfiles argument, 6 – 10
maxlogfiles parameter, default and max value, A – 19
maxloghistory parameter, default and max value, A – 19
maxlogmembers parameter, default and max value, A – 19
migrating, where to begin, 2 – 2
migrating from V5.1, 1 – 26
migration utility, when to use, 1 – 26
minextents parameter, maximum value, A – 19
monitor latch, display, 6 – 8
monitoring performance, 6 – 10
mts_max_dispatchers, default value, A – 12
mts_max_servers parameter, default value, A – 12
mts_servers, default value, A – 12
multi-block read, setting, 6 – 7
MultiProtocol Interchange, requirements, 1 – 10
Multi-threaded server, initialization parameters, A – 12

N

Navigator, preinstallation, 2 – 30
network, SQL*Net V2 directory, 2 – 29
network file server, 6 – 5
next parameter, maximum value, A – 19
NFS, 6 – 5

- NLS_LANG, set for installation, 5 – 28
- nls_language parameter, default value, A – 12
- nls_territory parameter, default value, A – 12
- NO_MAKE, set for installation, 5 – 28
- NONAMESERVER, 2 – 28
- numwidth, range of values, A – 18

O

- obsolete products, ODL, 1 – 36
- OFA. *See* Oracle Base Structure
- office automation products, preinstallation, 2 – 34
- online documentation, 1 – 23
- online help, Installer, 3 – 8
- operating system
 - reference documents, 1 – 23
 - when to upgrade, 4 – 11
- ops\$ login
 - and the root user-id, 1 – 31
 - and UNIX command interpreter, 1 – 31
- Optimal Flexible Architecture. *See* Oracle Base Structure
- optimal parameter, maximum value, A – 19
- ORACLE
 - automatic startup and shutdown, 3 – 24
 - deinstall old distribution, 4 – 4
 - network configuration, 6 – 5
 - preparing to install, 2 – 21
 - restarting cleanly, 3 – 26
 - shutting down cleanly, 3 – 26
 - supported configurations, A – 13
- Oracle Base Structure, introduction, 2 – 4
- ORACLE environment variables
 - APIPATH, 1 – 26
 - NLS_LANG, 5 – 28
 - NO_MAKE, 5 – 28
 - ORACLE_HOME, 5 – 28
 - ORACLE_PATH, 5 – 28
 - ORACLE_SID, 5 – 28
 - ORAMAIL_EDITOR, 11 – 2
 - SRW_TMP, 7 – 6
 - TMPDIR, 5 – 30
- Oracle Mail, preinstallation, 11 – 2

- Oracle XA Library, updates, 12 – 2
- Oracle XA library
 - required libraries, 12 – 2
 - required public information, 12 – 2
- Oracle*Mail
 - installation messages, 3 – 21
 - preinstallation, 2 – 34
 - requirements, 1 – 12
- Oracle*Mail, API applications, 11 – 2
- Oracle*Terminal, requirements, 1 – 7
- ORACLE_BASE, set for installation, 5 – 27
- ORACLE_HOME
 - default value, WYSE, 1 – 26
 - set for installation, 5 – 28
- ORACLE_LPARGS, default value, 1 – 26
- ORACLE_LPPROG, default value, 1 – 26
- ORACLE_LPSTAT, default value, 1 – 26
- ORACLE_PAGER, default value, 1 – 26
- ORACLE_PATH
 - default value, 1 – 26
 - set for installation, 5 – 28
- ORACLE_SID, set for installation, 5 – 28
- ORACLE7 Server, upgrading, 4 – 11
- oraenv, 3 – 23
- ORAMAIL_EDITOR, 11 – 2
- orasrv process, where started from, 1 – 28
- os_authent_prefix, default value, A – 12
- osh shell, 1 – 33

P

- paging, 6 – 10
- parameters
 - async_write, 6 – 7
 - cpu_count, 6 – 8
 - create controlfile, default and max values, A – 19
 - create database, default and max values, A – 19
 - db_block_compute_checksums, 6 – 6
 - db_block_size, 6 – 9
 - db_block_write_batch, 6 – 7
 - db_file_multiblock_read_count, 6 – 7
 - db_writers, 6 – 6

- init.ora file, 3 – 22
- latch_wait_mult, 6 – 8
- latch_wait_time, 6 – 8
- log archiving, 6 – 9
- log_simultaneous_copies, 6 – 8
- obsolete, 6 – 7
- shared memory, 6 – 2
- spin_count, 6 – 8
- storage, maximum values, A – 19
- tuning with, 6 – 6
- use_readv, 6 – 7
- pctincrease parameter, maximum value, A – 19
- post processing of terminal I/O, problems, 1 – 35
- post-installation activities, 3 – 22
- preinstallation, recommendations. *See* Oracle Base Structure
- preinstallation activities, 2 – 21
- Pro*Ada
 - linking, 8 – 2
 - preinstallation, 2 – 25
 - requirements, 1 – 8
 - updates, 8 – 2
- Pro*C
 - add to PATH, 8 – 3
 - preinstallation, 2 – 25
 - requirements, 1 – 8
 - updates, 8 – 3
- Pro*COBOL
 - COBDIR, 1 – 30
 - COBDIR variable, 8 – 4
 - preinstallation, 2 – 25
 - requirements, 1 – 8
 - sample program, 8 – 4
 - updates, 8 – 4
 - upgrading, 2 – 25
- Pro*FORTRAN
 - preinstallation, 2 – 26
 - requirements, 1 – 8
 - updates, 8 – 6
 - upgrading, 2 – 26
- problems, raw devices, 6 – 15
- processes, concurrent, limit to number of, 1 – 28
- processes parameter, default value, A – 12
- .profile file, 3 – 23
- ps tool, 6 – 11

R

- ranlib process, messages, 1 – 28
- raw devices
 - advantages, 6 – 14
 - backing up, 6 – 21
 - criteria for using, 6 – 16
 - problems, 6 – 15
 - setting up, 6 – 19
 - tuning with, 6 – 20
- raw input mode, reset terminal, 1 – 27
- readme.dat file, contents, 3 – 4
- README.doc file, contents, 3 – 4
- readmeunix.doc file, contents, 3 – 4
- redo log files, default names and sizes, 2 – 23
- relinking
 - discussion, 5 – 30
 - space considerations, 5 – 30
 - when you must, 5 – 30
- requirements
 - disk space and memory, 1 – 13
 - for ORACLE
 - CD-ROM device, 1 – 4
 - Compilers, 1 – 5
 - controller, 1 – 4
 - CPU, 1 – 4
 - memory, 1 – 4
 - Networking software, 1 – 5
 - terminal, 1 – 4
 - for Oracle, operating system, 1 – 4
- MultiProtocol Interchange, 1 – 10
- Oracle*Mail, 1 – 12
- Oracle*Terminal, 1 – 7
- Pro*Ada, 1 – 8
- Pro*C, 1 – 8
- Pro*COBOL, 1 – 8
- Pro*FORTRAN, 1 – 8
- SQL*Forms V3.0, 1 – 6
- SQL*Menu V5.0, 1 – 6

- SQL*Net V1
 - Asynchronous, 1-9
 - DECnet, 1-9
 - TCP/IP, 1-9
- SQL*Net V2, 1-10
 - DECnet, 1-10
 - SPX/IPX, 1-10
 - TCP/IP, 1-11
- SQL*Plus, 1-7
- SQL*Report, 1-7
- SQL*ReportWriter, 1-7
- resource limits, 6-25
- restrictions
 - generic UNIX, 1-31
 - SQL*Plus with ORACLE7, 1-33
 - SQL*ReportWriter, 7-6
- return codes, UNIX restriction, 1-32
- root user, and ops\$ login, 1-31
- root.sh, running, 3-22
- rowid component lengths, A-19

S

- sar tool, options, 6-23
- screen tools, restrictions, 1-29
- scripts
 - .SQL, A-10
 - run automatically, 3-27
- SGA
 - init.ora parameters, 3-22
 - latches, 6-8
 - size limits, 1-22
- shared memory
 - discussion, 1-22
 - operations. *See* SGA, size
 - options, 1-22
 - parameter adjustments, 6-2
- shared_pool_size parameter, default value, A-12
- SHELL, 1-23
 - default value, WYSE, 1-26
- shell, correct one for Oracle scripts, 1-23
- SHELL variable, workaround, 1-35
- shutdown, errors, 1-33

- size
 - database file, 1-25
 - log file, 1-25
 - SGA, 1-22
- sort_area_size parameter, default value, A-12
- sort_spacemap_size parameter, default value, A-12
- spin_count parameter, 6-8
- SPOOL command, in SQL*Plus, 7-2
- spoolcmd, SQL*ReportWriter, 7-6
- SPX/IPX Adapter, updates, 10-3
- SQL scripts, locations of, A-10
- SQL*DBA, A-17
- SQL*Forms, installation messages, 3-18
- SQL*Forms V3.0
 - requirements, 1-6
 - updates, 7-3
- SQL*Loader, converting from ODL, 1-36
- SQL*Menu, installation messages, 3-18
- SQL*Menu V5.0
 - requirements, 1-6
 - updates, 7-5
- SQL*Net Asynchronous
 - deinstalling, 2-27
 - setup, 2-27
 - sqlnet account requirement, 2-27
 - updates, 1-30
 - user account, 4-15
 - sqlnet, 4-15
- SQL*Net DECnet
 - updates, 1-30
 - user account, 4-15
 - dni, 4-15
- SQL*Net TCP/IP, setup, 2-28
- SQL*Net V1
 - Asynchronous, requirements, 1-9
 - DECnet, requirements, 1-9
 - linking specific drivers, 9-2
 - overview, 9-2
 - TCP/IP, requirements, 1-9
 - updates, 1-30
- SQL*Net V2
 - DECnet, requirements, 1-10
 - installation messages, 3-20
 - installing, 10-2

- post-installation, 10 – 2, 10 – 3
- preinstallation, 2 – 29
- README.doc file location, 2 – 29
- requirements, 1 – 10
- SPX/IPX, requirements, 1 – 10
- TCP/IP, requirements, 1 – 11
- SQL*Plus
 - COPY command, 1 – 33
 - installation messages, 3 – 18
 - requirements, 1 – 7
 - setup files, glogin.sql, 7 – 2
 - updates, 7 – 2
 - version restriction with ORACLE7, 1 – 33
- SQL*Report, requirements, 1 – 7
- SQL*ReportWriter
 - installation messages, 3 – 18
 - printer specifications, 7 – 6
 - requirements, 1 – 7
 - restrictions, 7 – 6
 - spoolcmd, 7 – 6
 - temp file location, 7 – 6
- SQL*ReportWriter V1.0, updates, 7 – 6
- sqlnet. *See* SQL*Net Asynchronous
- SRW_TMP, 7 – 6
- startup files, 3 – 23
- storage parameters, maximum values, A – 19
- sun.r, and underlining, 1 – 29
- swap space, restriction, 1 – 28
- system requirements, for ORACLE, 1 – 4

T

- tables, estimating size of, A – 10
- tape archival, workaround, 1 – 29
- temporary_table_locks, default value, A – 12
- terminal, resetting, 1 – 27
- terminal I/O, post processing problems, 1 – 35
- TK2DEV settings, Installer, 3 – 7
- TMPDIR, and relinking, 5 – 30
- TNS listener, setuid root, 10 – 2, 10 – 3
- tnslsnr, setuid root, 10 – 2, 10 – 3
- transactions_per_rollback_segment
 - default value, A – 12
 - restrictions, 1 – 32

- troubleshooting, Installer, 3 – 17
- tstshm, 6 – 2
 - sample output, 6 – 3
- tuning
 - CPU usage, 6 – 10
 - discussion of, 6 – 6
 - I/O activity, 6 – 10
 - latches, 6 – 8
 - monitoring performance, 6 – 10
 - paging, 6 – 10
 - tools for, 6 – 11
 - using raw devices, 6 – 20
- TWO_TASK, default value, WYSE, 1 – 26

U

- underlining, support in Oracle*Terminal files, 1 – 29
- UNIX
 - exit code errors, 1 – 32
 - import example syntax, 1 – 32
- upgrade, export, when to do, 1 – 26
- upgrade the operating system, when to do, 4 – 11
- upgrading, where to begin, 2 – 2
- upgrading ORACLE7, export upgrade, 4 – 15
- upgrading ORACLE7 Server, 4 – 11
 - preliminary operations, 4 – 15
- use_readv, 6 – 7
- user_dump_dest parameter, default value, A – 12

V

- variables, environmental, 1 – 26
- volume size, 4 – 13

W

- windows, resizing, 1 – 28
- work-arounds, generic UNIX, 1 – 31

X

xtermsun.r, and underlining, 1 – 29

Reader's Comment Form

ORACLE7 for Sun-SPARC SunOS 4.1.x Installation & Configuration Guide, Release 7.0.12
Part No. 7102-70-0193

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any suggestions for improvement, please indicate the topic, chapter, and page number below:

Please send or FAX your comments to:

Jane Sokolow
UNIX Technical Publications Manager
Oracle Corporation
500 Oracle Parkway
Box 659407
Redwood Shores, CA 94065
(415) 506-2456 FAX (415) 506-7223

If you would like a reply, please give your name, address, and telephone number below:

Thank you for helping us improve our documentation.

For Customer Technical Support Services, call: **(415) 506-1500**

ORACLE®

7102-70-0193